

JOKOWI'S WHITE PAPER

**KAJIAN
FORENSIK
DIGITAL**



**DR.ENG RISMON H.
SIANIPAR, S.T, M.T, M.ENG**



JOKOWI'S WHITE
PAPER:
Kajian Forensik

RISMON HASIHOLAN SIANIPAR
BALIGE PUBLISHING

DOWNLOAD GRATIS, CETAK SENDIRI, DAN SEBAR LUASKAN BUKU INI!

**JIKA KALIAN BERKENAN, BANTU UNTUK PROYEK PENULISAN
BUKU SELANJUTNYA (VINA CIREBON, KM 50, dll)**

Bank: BNI Rek: 090 - 215 - 721-0

An. Rismon Hasiholan Sianipar

KATA PENGANTAR

Perkembangan teknologi digital telah membuka peluang baru dalam bidang forensik, khususnya untuk menguji keaslian dokumen analog. Melalui metode digital forensic, berbagai detail yang tidak dapat dilihat dengan mata telanjang dapat dianalisis secara mendalam. Dokumen fisik seperti ijazah, sertifikat, maupun surat resmi dapat dipindai dan diperiksa berdasarkan pola cetak, warna, serta tipografi untuk membedakan antara dokumen asli dan dokumen palsu.

Dalam konteks penelitian ini, digital forensic digunakan untuk menyingkap indikasi bahwa sejumlah dokumen analog sejatinya berasal dari rekayasa digital. Analisis citra mampu mengidentifikasi pola jarak huruf yang tidak konsisten, keseragaman tinta yang mencurigakan, hingga tanda hasil cetak komputer yang berbeda dari cetakan manual. Perbandingan dengan dokumen asli menjadi kunci untuk mengungkap adanya manipulasi.

Buku ini menghadirkan beragam metode analisis, mulai dari uji lintasan stempel, Error Level Analysis (ELA), Local Binary Pattern (LBP), K-Means Color Clustering, hingga Multi-Scale Gradient Analysis. Setiap teknik forensik digital ini digunakan untuk memeriksa elemen penting dokumen seperti teks, tanda tangan, logo, foto, hingga stempel. Temuan-temuan yang dihasilkan menunjukkan adanya ketidakwajaran yang mengarah pada proses penyusunan digital, bukan pada metode cetak otentik.

Selain itu, analisis tipografi dan tekstur huruf memperlihatkan perbedaan jelas antara dokumen yang dicetak dengan mesin ketik atau letterpress dibandingkan dengan dokumen yang disusun menggunakan word processor modern. Konsistensi huruf, distribusi spacing, hingga pola

cetak mikro mampu mengungkap indikasi kuat adanya rekonstruksi atau pemalsuan. Dengan pendekatan ini, forensik digital membuktikan perannya sebagai instrumen penting dalam autentikasi dokumen.

Akhirnya, buku ini menunjukkan bahwa kombinasi metode forensik digital dapat memberikan bukti visual, tekstural, dan matematis yang konsisten untuk membedakan dokumen asli dengan yang dimanipulasi. Melalui studi kasus seperti analisis ijazah Joko Widodo, karya ini memperlihatkan bagaimana sains forensik digital dapat menjadi alat verifikasi objektif, tidak hanya dalam kajian akademik, tetapi juga dalam pembuktian hukum dan sejarah.

BALIGE, SUMUT, 1 SEPTEMBER 2025
PENULIS

RISMON HASIHOLAN SIANIPAR

Ringkasan

Digital forensic bisa dipakai untuk mengotentikasi dokumen analog palsu dengan cara memeriksa detail yang tidak terlihat jelas oleh mata manusia. Dokumen fisik seperti sertifikat, ijazah, atau surat resmi bisa difoto atau dipindai, lalu dianalisis secara teliti untuk melihat perbedaan pola cetak, warna, atau bentuk huruf yang tidak wajar. Dari sini, ahli forensik bisa membedakan apakah dokumen tersebut benar-benar dibuat dengan cara dan alat yang sesuai zaman, atau justru dibuat belakangan dengan teknologi modern.

Selain itu, digital forensic juga bisa mengungkap tanda-tanda bahwa sebuah dokumen analog sebenarnya berasal dari proses pembuatan digital. Misalnya, ada kesalahan penempatan huruf, jarak antarbaris yang tidak konsisten, atau tanda-tanda hasil cetak komputer yang berusaha ditiru menjadi dokumen lama. Dengan membandingkan hasil pemeriksaan ini dengan dokumen asli sebagai pembanding, ahli dapat menentukan apakah sebuah dokumen analog memang asli atau hanya salinan palsu yang dibuat untuk menipu.

Pada buku ini, Uji lintasan stempel pada ijazah Jokowi dilakukan untuk melihat bagaimana tinta stempel berinteraksi dengan latar belakang dokumen. Dalam analisis digital, area yang dilewati stempel diperiksa berdasarkan nilai intensitas warna, termasuk pada kanal merah (red channel). Hasilnya menunjukkan bahwa nilai intensitas pada kanal merah mendekati nol, yang berarti hampir tidak ada komponen warna merah di bagian yang terkena stempel. Ini menunjukkan tinta stempel sepenuhnya menyerap cahaya merah, ciri khas tinta berwarna biru atau ungu tua yang pekat.

Temuan ini penting karena dapat dipakai untuk menguji keaslian proses penempelan stempel. Stempel asli biasanya meninggalkan variasi warna dan ketebalan tinta akibat tekanan tangan yang tidak merata, sehingga

nilai intensitasnya akan berfluktuasi. Namun, bila nilai kanal merah konsisten sangat rendah dan lintasannya terlalu rata, hal ini dapat menjadi indikasi bahwa stempel tersebut bukan hasil penempelan fisik langsung, melainkan hasil cetak atau tempelan digital di atas dokumen. Dengan demikian, uji ini membantu membedakan antara stempel asli dan stempel yang dimanipulasi.

Selanjutnya, hasil analisis Error Level Analysis (ELA) dengan teknik CLAHE pada pada buku ini dilakukan terhadap gambar ijazah Jokowi memperlihatkan kontras yang jauh lebih tinggi di bagian-bagian tertentu dibandingkan latar belakang kertas. CLAHE membantu memperjelas perbedaan tingkat kompresi pada dokumen, sehingga anomali seperti pada teks “IJAZAH”, lambang UGM, tanda tangan, foto, hingga cap meterai terlihat sangat jelas. Hampir semua elemen penting tersebut menampilkan highlight merah yang padat dan seragam, menandakan bahwa profil kompresinya berbeda dari kertas asli. Perbedaan ini mengindikasikan kemungkinan kuat bahwa elemen-elemen tersebut tidak berasal dari satu proses pemindaian tunggal, melainkan hasil penyisipan atau pengeditan digital.

Selain itu, hasil analisis juga menunjukkan ketidakwajaran pada teks tanggal, lokasi penerbitan, dan beberapa blok tulisan penting yang memiliki struktur digital berbeda dari teks sekitarnya. Latar belakang kertas justru tampak alami dan konsisten, sementara objek-objek seperti stempel, tanda tangan, dan foto terlihat “menempel” dengan tepi yang sangat tajam tanpa transisi bayangan alami. Pola ini, ditambah adanya artefak penyuntingan seperti edge enhancement, memperkuat dugaan bahwa dokumen merupakan kompilasi dari berbagai elemen digital yang digabungkan, bukan hasil cetakan fisik yang kemudian dipindai secara utuh.

Selanjutnya pada buku ini dilakukan analisis dengan metode LBP (Local Binary Pattern) pada ijazah menunjukkan perbedaan mencolok antara dokumen milik Frono Jiwo dan Joko Widodo. Pada ijazah Frono Jiwo, sejumlah huruf seperti “A”, “O”, “I”, “D”, dan “E” memiliki tingkat tumpang tindih yang tinggi dengan logo, bahkan ada yang mencapai lebih dari 40%.

Fenomena ini umum ditemukan pada dokumen resmi karena segel atau logo memang sengaja dicetak menindih teks sebagai bentuk pengamanan. Tekstur huruf yang terganggu oleh segel menunjukkan bahwa elemen-elemen tersebut dicetak bersamaan, tanpa upaya untuk menghindari tumpang tindih, yang justru menjadi tanda keaslian.

Sebaliknya, pada ijazah Joko Widodo, tingkat overlapping pada huruf-huruf di dekat logo sangat rendah, bahkan ada yang 0%. Meskipun tampak “bersih” secara visual, ketiadaan tumpang tindih ini dalam forensik dokumen justru mengindikasikan bahwa logo ditempatkan secara digital setelah teks dibuat. Tekstur huruf yang seragam tanpa gangguan dari logo, serta posisi segel yang sepenuhnya menghindari teks, menghilangkan ciri khas overprinting pada dokumen asli. Hal ini memperkuat dugaan bahwa dokumen tersebut telah direkayasa atau dicetak ulang dari hasil penyusunan digital, bukan berasal dari proses cetak resmi yang dilakukan sekaligus.

Kemudian, analisis dengan K-Means Color Clustering pada huruf-huruf yang berada di atas logo ijazah mengungkap perbedaan urutan pencetakan antara dokumen kiri (Frono Jiwo) dan kanan (Joko Widodo). Pada ijazah Frono Jiwo, warna dan tekstur logo terlihat masuk ke dalam bentuk huruf, mengganggu konsistensi warna hitam dan membentuk pola tumpang-tindih alami. Hal ini menunjukkan bahwa teks dicetak lebih dulu, lalu logo dicap di atasnya, sebagaimana prosedur cetak resmi yang umum pada dokumen otentik. Sebaliknya, pada ijazah Joko Widodo, huruf-huruf terlihat bersih, solid, dan bebas gangguan warna dari logo di bawahnya. Ini mengindikasikan bahwa teks kemungkinan ditempatkan di atas logo menggunakan proses digital, bukan hasil pencetakan fisik bersamaan.

Ciri lain yang memperkuat temuan ini adalah bentuk huruf pada dokumen kiri yang menampilkan degradasi di pinggiran akibat tekanan atau tekstur cap logo, sedangkan pada dokumen kanan, batas huruf terlalu sempurna tanpa interaksi visual dengan latar belakang. Rasio piksel warna gelap pada huruf di dokumen kiri juga lebih kecil karena sebagian tertutupi logo, sementara di dokumen kanan rasio tersebut

besar dan konsisten. Kombinasi bukti ini menunjukkan bahwa dokumen kiri (Frono Jiwo) mencerminkan proses pencetakan otentik, sedangkan dokumen kanan (Joko Widodo) memiliki tanda kuat manipulasi digital di mana teks ditambahkan belakangan di atas logo.

Selanjutnya, analisis OCR berbasis kontur dan histogram spasial memperlihatkan perbedaan jelas antara dokumen dengan huruf proporsional (kolom kiri, lembar pengesahan pembimbing skripsi Joko Widodo) dan huruf monospasi (kolom kanan, lembar pengesahan pembimbing skripsi Budi Darmito). Dokumen di kolom kiri menunjukkan variasi jarak antarkhuruf yang lebar, deviasi standar tinggi, dan distribusi spacing yang bergantung pada bentuk karakter—ciri khas font digital dengan kerning. Sebaliknya, dokumen di kolom kanan memiliki jarak huruf yang konsisten, deviasi sangat rendah, dan histogram dengan satu puncak dominan, yang merupakan tanda khas hasil ketikan mesin tik. Perbedaan ini konsisten pada semua sampel teks, baik yang pendek maupun panjang.

Temuan ini memiliki nilai forensik yang kuat, karena dokumen yang mengklaim berasal dari mesin tik atau letterpress seharusnya menampilkan pola monospasi yang seragam. Jika pola spacing menunjukkan karakteristik proporsional, sangat kuat indikasi bahwa dokumen tersebut dibuat dengan word processor modern. Dalam konteks pembuktian hukum, metode ini dapat mengungkap pemalsuan atau rekonstruksi dokumen, bahkan jika telah dicetak ulang atau difoto. Hasil dari analisis ini memperlihatkan kontras tajam antara ciri dokumen digital dan dokumen mesin tik atau letterpress, membuktikan bahwa tipografi forensik adalah alat efektif untuk autentikasi dokumen.

Selanjutnya, dilakukan perbandingan visual dan analisis difference map berbasis descriptor SIFT (Scale Invariant Feature Transform) yang menunjukkan tingkat kemiripan bentuk sejumlah huruf dan kata pada kedua dokumen yang luar biasa tinggi, mencapai 90,57%. Kesamaan ini meliputi proporsi, ketebalan garis, jarak antarkhuruf, hingga detail kecil seperti titik pada huruf “i” dan bentuk counter space. Ciri khas letterpress manual seperti variasi ketebalan tinta, ketidaksejajaran huruf, atau

tekstur akibat tekanan cetak tidak ditemukan. Sebaliknya, kedua kata menampilkan konsistensi dan presisi yang identik dengan output font digital modern, mengindikasikan bahwa keduanya kemungkinan berasal dari sumber file yang sama.

Dari perspektif forensik dokumen, kesamaan pixel-to-pixel ini hampir mustahil dicapai jika satu dokumen benar-benar dicetak dengan handpress dan yang lain dengan printer digital. Tekstur tepi huruf pada dokumen kiri juga memperlihatkan tanda-tanda antialiasing khas cetakan digital, bukan penyebaran tinta alami pada serat kertas. Dengan demikian, bukti visual, tekstural, dan numerik secara konsisten menunjukkan bahwa dokumen lembar pengesahan pembimbing skripsi Joko Widodo bukanlah hasil cetakan letterpress asli, melainkan reproduksi digital dari dokumen kanan atau sumber digital yang sama, kemungkinan besar diketik menggunakan Microsoft Word atau perangkat lunak serupa.

Selanjutnya, Analisis berbasis metode Multi-Scale Gradient Analysis dilakukan untuk mendeteksi konsistensi lintasan visual objek seperti stempel pada berbagai skala, yang sangat efektif dalam memeriksa keaslian dokumen resmi seperti ijazah. Pada kasus ijazah Joko Widodo yang dianalisis, stempel merah tampak melintasi foto, materai, dan bidang kosong kertas, namun hasil visualisasi gradien menunjukkan ketidaksesuaian pada area foto. Di bagian kertas dan materai, tepi stempel terlihat kuat dan jelas, sedangkan pada bagian foto tepi tersebut jauh lebih lemah atau bahkan hilang. Kondisi ini mengindikasikan bahwa stempel tidak benar-benar menempel di atas foto, melainkan kemungkinan foto tersebut ditempel atau disisipkan secara digital setelah proses penyegelan. Hal ini diperkuat oleh hilangnya pola warna kompleks pada visualisasi gradien multi-warna di area foto, serta hasil deteksi tepi yang tidak menunjukkan lintasan stempel yang menyatu secara alami melintasi wajah dan pakaian.

Temuan ini tetap konsisten meskipun parameter pengolahan citra seperti Gaussian Blur, CLAHE, dan threshold pada Sobel filter disesuaikan. Respons gradien stempel di luar foto tetap jelas, sementara di atas foto

tetap lemah, yang bertolak belakang dengan karakteristik stempel asli yang seharusnya konsisten di seluruh lintasan. Fakta bahwa materai terintegrasi dengan stempel secara alami sementara foto tidak, semakin memperkuat dugaan adanya manipulasi digital atau penempelan fisik foto setelah stempel dibubuhkan. Berdasarkan bukti visual dan matematis dari analisis ini, terdapat indikasi kuat bahwa lintasan stempel pada dokumen tersebut telah mengalami intervensi non-asli, menjadikan Multi-Scale Gradient Analysis sebagai alat krusial dalam deteksi pemalsuan dokumen.

Selanjutnya, pada uji glyph, analisis huruf “G” dari dokumen kiri (lembar pengesahan skripsi Jokowi) dan dokumen kanan (rekonstruksi Microsoft Word) dilakukan dengan dua metode: ORB (Oriented FAST and Rotated BRIEF) dan template matching. ORB mendeteksi fitur-fitur lokal seperti sudut dan lengkungan, menghasilkan skor 36,23% yang relatif sedang karena sensitif terhadap noise, ketebalan tinta, dan degradasi pemindaian. Meski begitu, visualisasi ORB menunjukkan banyak titik kunci yang cocok di sepanjang kontur huruf, menandakan struktur geometrisnya sama. Sementara itu, template matching yang membandingkan pola keseluruhan huruf menghasilkan skor 83,53%, angka tinggi yang menunjukkan bentuk huruf hampir identik. Nilai gabungan sebesar 50,42% semakin memperkuat kesimpulan bahwa huruf dari kedua dokumen memiliki kemiripan signifikan.

Secara visual, perbesaran huruf “G” menunjukkan kesamaan lengkungan atas, batang vertikal kiri, dan spur bawah pada kedua dokumen. Perbedaan ketajaman tepi huruf di dokumen kiri hanyalah akibat cetakan fisik dan proses digitalisasi, bukan perbedaan font. Garis-garis kecocokan ORB membuktikan konsistensi geometri huruf, sedangkan skor template matching yang tinggi menunjukkan siluet huruf sama persis. Semua ciri khas font tetap terjaga sehingga kecil kemungkinan huruf berasal dari font berbeda atau dibuat manual. **Dengan demikian, huruf “G” pada dokumen kiri identik dengan huruf “G” pada dokumen kanan.**

Terakhir, metode Noise Pattern Analysis (NPA) berbasis Local Binary Pattern (LBP) digunakan untuk membedakan jenis teknologi pencetakan melalui analisis tekstur mikro pada dokumen. Pada kasus lembar pengesahan skripsi yang diklaim dicetak manual, hasil analisis tiga area teks menunjukkan pola granular khas printer inkjet. Peta variansi lokal (LocalVar) menampilkan butiran tinta acak dengan nilai tinggi (2817.02, 1657.84, dan 1860.55), sementara citra LBP memperlihatkan bayangan titik-titik tinta individual. Histogram LBP juga menunjukkan distribusi yang menyebar dengan puncak di bin pertengahan, berbeda dengan letterpress yang biasanya menghasilkan tekstur solid dan histogram tajam.

Konsistensi temuan di seluruh area menegaskan bahwa dokumen tidak dicetak menggunakan handpress atau letterpress, karena tidak ada ciri khas mekanis seperti cetakan timbul, tekanan fisik, atau tepi tinta yang rata dan tajam. Sebaliknya, pola mikro menunjukkan distribusi acak dan homogen khas printer inkjet. Dengan demikian, metode NPA berbasis LBP berhasil membuktikan secara objektif bahwa dokumen tersebut sepenuhnya dicetak dengan teknologi inkjet, membantah klaim cetakan manual.

DAFTAR ISI

Bab 1	Pengantar - Digital Forensics Untuk Autentikasi Dokumen Analog Palsu - Kasus Pemalsuan Hitler's Diaries - Kasus Pemalsuan Killian Document	1 1 4 9
Bab 2	Polemik Proses Akademik Joko Widodo - Transkrip Akademik Tanpa Tanda Tangan Otoritas UGM - Skripsi Tanpa Tanda Tangan Penguji - Joko Widodo Terdaftar Program Sarjana Muda - Bantahan Kasmudjo Dalam Perannya Membimbing Joko Widodo - Potensi Rekayasa Lembar Pengesahan Skripsi	15 15 19 22 27 28
Bab 3	Uji Lintasan Stempel - Analisa Sebaran Intensitas Rgb Pada Foto Ijazah Jokowi - Skrip Program Python Analisa Ruang Warna RGB - Lintasan Stempel Merah Terdeteksi Pada Ruang Warna RGB - Lintasan Stempel Merah Tidak Terdeteksi Pada Ruang Warna RGB - Analisa Sebaran Intensitas Ruang Warna CMYK (Cyan, Magenta, Yellow, Black) Pada Foto Ijazah Jokowi - Skrip Program Python Analisa Ruang Warna CMYK - Lintasan Stempel Merah Tidak Terdeteksi Pada Ruang Warna CMYK - Lintasan Stempel Merah Tidak Terdeteksi Pada Ruang Warna CYMK - Lintasan Stempel Merah Terdeteksi Pada Ruang Warna HSV	33 33 42 45 49 55 59 63 69 73
Bab 4	Uji Ela (Error Level Analysis) - Tentang ELA (Error Level Analysis) - Implementasi ELA (Error Level Analysis) - Hasil Implementasi ELA Atas Ijazah Jokowi - Implementasi ELA Dengan Adaptive Brightness Scaling	75 75 79 85 89

	• Hasil Implementasi ELA Dengan Adaptive Brightness Scaling Atas Ijazah Jokowi • Implementasi ELA Dengan CLAHE (Contrast Limited Adaptive Histogram Equalization) • Hasil Implementasi ELA Dengan CLAHE (Contrast Limited Adaptive Histogram Equalization) Atas Ijazah Jokowi	95 100 106
Bab 5	Overlapping Detection • Tentang Overlapping Detection • Implementasi Overlapping Detection Pada Ruang Warna HSV (Hue, Saturation, And Value) • Hasil Implementasi Overlapping Detection Pada Ruang Warna HSV • Implementasi Overlapping Detection Dengan Adaptive Thresholding • Hasil Implementasi Overlapping Detection Dengan Adaptive Thresholding • Implementasi Overlapping Detection Dengan LBP (Local Binary Pattern) • Hasil Implementasi Overlapping Detection Dengan LBP (Local Binary Pattern) • Implementasi Overlapping Detection Dengan Canny Edge Detector Dan Dilation • Hasil Implementasi Overlapping Detection Dengan Canny Edge Detector Dan Dilation	112 112 114 122 125 133 138 145 149 156
Bab 6	Analisis Lapisan Visual • Tentang Analisis Lapisan Visual • Analisis Lapisan Visual Dengan K-Means Clustering • Hasil Analisis Lapisan Visual Dengan K-Means Clustering • Analisis Lapisan Visual Dengan Segmentasi Warna HSV (Hue, Saturation, Value) • Hasil Analisis Lapisan Visual Dengan Segmentasi Warna HSV (Hue, Saturation, Value) • Analisis Lapisan Visual Dengan GMM (Gaussian Mixture Model) • Hasil Analisis Lapisan Visual Dengan GMM (Gaussian Mixture Model)	162 162 166 172 177 184 188 196
Bab 7	Uji Proportional Spacing dan Kerning • Tentang Proportional Spacing Dan Kerning • Implementasi Proportional Spacing Dan Kerning Dengan OCR (Optical Character Recognition) • Hasil Implementasi Proportional Spacing Dan Kerning Dengan OCR • Implementasi Proportional Spacing Dan Kerning Dengan OCR (Optical Character Recognition) Dan Edge Detection	201 201 206 213 218

	• Hasil Implementasi Proportional Spacing Dan Kerning Dengan OCR (Optical Character Recognition) Dan Edge Detection • Implementasi Proportional Spacing Dan Kerning Dengan OCR (Optical Character Recognition), Edge Detection, Dan Morfologi Citra • Hasil Implementasi Proportional Spacing Dan Kerning Dengan OCR (Optical Character Recognition), Edge Detection, Dan Morfologi Citra	226 231 240
Bab 8	Uji Overlay • Tentang Overlay • Implementasi Overlay Difference Map • Hasil Implementasi Overlay Difference Map • Implementasi Overlay Difference Map Berbasis SIFT (Scale-Invariant Feature Transform) • Hasil Implementasi Overlay Difference Map Berbasis SIFT • Implementasi Overlay Difference Map Berbasis BRISK (Binary Robust Invariant Scalable Keypoints) • Hasil Implementasi Overlay Difference Map Berbasis BRISK	247 247 250 259 277 290 303 315
Bab 9	Gradient Analysis • Tentang Gradient Analysis • Implementasi Gradient Magnitude Analysis • Hasil Implementasi Gradient Magnitude Analysis • Implementasi Directional Gradient Mapping • Hasil Implementasi Directional Gradient Mapping • Implementasi Directional Multi-Scale Gradient Analysis • Hasil Implementasi Directional Multi-Scale Gradient Analysis	330 330 332 344 350 355 360 366
Bab 10	Analisa Glyph • Tentang Analisa Glyph • Analisa Glyph Dengan Deskriptor ORB (Oriented Fast and Rotated Brief) Dan Ssim (Structural Similarity Index) • Hasil Analisa Glyph Dengan Deskriptor ORB Dan SSIM • Analisa Glyph Dengan Deskriptor ORB (Oriented Fast and Rotated Brief) Dan HOG (Histogram Of Oriented Gradients) • Hasil Analisa Glyph Dengan Deskriptor ORB dan HOG • Analisa Glyph Dengan Deskriptor ORB (Oriented Fast and Rotated Brief) Dan Template Matching • Hasil Analisa Glyph Dengan Deskriptor ORB Dan Template Matching	372 372 379 391 397 409 415 422
Bab 11	NOISE PATTERN ANALYSIS • Tentang Noise Pattern Analysis • Implementasi Noise Pattern Analysis Berbasis FFT (Fast Fourier Transform)	430 430 436

	• Hasil Noise Pattern Analysis Berbasis FFT • Implementasi Noise Pattern Analysis Berbasis HOG (Histogram Of Oriented Gradients) • Hasil Noise Pattern Analysis Berbasis HOG • Implementasi Noise Pattern Analysis Berbasis LBP (Local Binary Pattern) • Hasil Noise Pattern Analysis Berbasis LBP	446 453 464 470 481
	Referensi	487

DIGITAL FORENSICS UNTUK AUTENTIKASI DOKUMEN ANALOG PALSU

SEJARAH SINGKAT



Digital forensics adalah cabang dari ilmu forensik yang berkaitan dengan identifikasi, pelestarian, analisis, dan presentasi bukti digital yang dapat digunakan dalam proses hukum. Meskipun awalnya berfokus pada perangkat digital seperti komputer dan ponsel, dalam perkembangannya digital forensics juga memainkan peran penting dalam

autentikasi dokumen analog, terutama ketika dokumen tersebut didigitalkan untuk analisis. Peran ini menjadi krusial dalam kasus pemalsuan dokumen sejarah, kontrak, sertifikat, hingga surat resmi yang berbentuk fisik namun dikaji melalui pendekatan digital.

Autentikasi dokumen analog palsu melalui digital forensics melibatkan proses konversi dokumen fisik menjadi bentuk digital, seperti melalui pemindaian (scanning) resolusi tinggi. Setelah dokumen berada dalam bentuk digital, berbagai teknik pengolahan citra dan analisis forensik dapat diterapkan untuk meneliti keasliannya. Ini mencakup analisis tekstur kertas, tinta, pola cetak, dan artefak digital yang muncul dalam proses pemindaian. Digital forensics dalam konteks ini berfokus pada jejak visual dan metadata tersembunyi yang dapat menunjukkan adanya modifikasi atau pemalsuan.

Salah satu pendekatan penting adalah analisis font dan tipografi. Dokumen historis atau resmi memiliki karakteristik font tertentu sesuai dengan periode waktu atau teknologi yang digunakan saat itu. Melalui perbandingan digital, penyidik dapat mendeteksi penggunaan font yang tidak sesuai dengan zaman pembuatan dokumen. Contohnya adalah kasus "Hitler Diaries" di mana digital forensics membantu mengidentifikasi jenis tinta dan metode pencetakan yang tidak sesuai dengan periode Nazi.

Teknik image forensics juga memegang peranan penting. Dengan menganalisis noise (gangguan visual), kompresi JPEG, dan ketidaksesuaian bayangan atau struktur huruf, penyidik dapat mendeteksi apakah sebagian isi dokumen telah diedit atau ditambahkan. Misalnya, ketika satu bagian teks memiliki karakteristik noise yang berbeda dari bagian lain, hal itu dapat mengindikasikan penyisipan gambar atau teks palsu secara digital ke dalam pemindaian dokumen analog.

Dalam beberapa kasus, eksplorasi metadata pemindaian juga dapat memberikan petunjuk penting. Beberapa pemindai menyimpan informasi teknis ke dalam file gambar, seperti model pemindai, waktu pemindaian, dan parameter pemrosesan. Inkonsistensi dalam metadata ini dapat menjadi indikator adanya manipulasi file digital yang merepresentasikan dokumen analog tersebut.

Selain itu, teknik pencocokan inkonsistensi pencahayaan dan arah bayangan digunakan untuk mengungkap kejanggalan visual yang tidak

dapat dengan mudah dilihat oleh mata manusia. Misalnya, jika sebagian teks dalam dokumen memiliki arah bayangan yang berbeda, hal ini bisa menandakan bahwa elemen tersebut telah ditempelkan secara digital dari sumber lain. Teknologi ini diperkuat dengan algoritma pembelajaran mesin yang dilatih untuk mengenali pola-pola manipulasi visual.

Digital forensics juga melibatkan analisis spektral terhadap dokumen yang dipindai. Melalui pencitraan multispektral atau hiperspektral, forensik digital dapat mengidentifikasi perbedaan kimia pada tinta atau bahan kertas, bahkan jika perbedaan itu tidak terlihat di bawah cahaya tampak biasa. Pendekatan ini sering digunakan oleh lembaga arsip dan museum untuk memastikan keaslian dokumen sejarah.

Peran digital forensics dalam autentikasi dokumen analog juga mencakup penggunaan algoritma deteksi watermark. Dokumen resmi atau arsip bersejarah kadang memiliki watermark yang tercetak halus dalam struktur kertas. Dengan bantuan pengolahan citra digital, watermark ini dapat diperjelas dan dibandingkan dengan referensi asli untuk membuktikan keaslian atau mendeteksi pemalsuan.

Dalam konteks hukum dan investigasi, digital forensics tidak hanya membantu mengidentifikasi pemalsuan, tetapi juga memberikan bukti kuat yang dapat dipertanggungjawabkan di pengadilan. Hasil analisis digital biasanya disertai dengan laporan teknis yang memuat langkah-langkah analisis, metode yang digunakan, serta visualisasi temuan. Hal ini sangat membantu jaksa, pengacara, atau hakim dalam memahami sifat manipulasi yang terjadi pada dokumen.

Kesimpulannya, digital forensics memainkan peran vital dalam otentikasi dokumen analog palsu, terutama di era modern di mana banyak dokumen analog direpresentasikan dalam bentuk digital. Dengan menggabungkan pengolahan citra, analisis metadata, spektrum warna, dan kecerdasan buatan, digital forensics memungkinkan penyelidikan untuk mendeteksi manipulasi secara akurat, bahkan pada dokumen fisik yang tampaknya otentik. Teknologi ini terus berkembang seiring meningkatnya kompleksitas pemalsuan dokumen di era digital.

KASUS PEMALSUAN HITLER'S DIARIES



Kasus pemalsuan Hitler's Diaries adalah salah satu skandal sejarah dan jurnalistik terbesar di abad ke-20. Pada tahun 1983, majalah berita terkemuka Jerman, Stern, mengklaim telah memperoleh 62 volume buku harian yang ditulis oleh Adolf Hitler, yang dikatakan ditemukan dari reruntuhan pesawat Nazi pada akhir Perang Dunia II. Buku harian ini dijanjikan mengandung wawasan pribadi Hitler tentang perang, hubungannya dengan para petinggi Nazi, dan

keyakinan ideologisnya. Publikasi ini menciptakan kehebohan besar secara global karena dianggap sebagai penemuan dokumen sejarah yang sangat langka dan berharga.

Namun, setelah publikasi awal dan antusiasme besar dari media, berbagai ahli sejarah, termasuk Hugh Trevor-Roper (seorang sejarawan terkemuka Inggris), mulai meragukan keaslian dokumen tersebut. Trevor-Roper awalnya mendukung keasliannya, tetapi dalam hitungan hari setelah meninjau lebih lanjut, ia menarik kembali pendapatnya. Ini memicu penyelidikan lebih lanjut yang mengarah pada pengungkapan bahwa buku harian tersebut adalah pemalsuan yang dibuat oleh Konrad Kujau, seorang penipu asal Jerman Barat yang sebelumnya telah dikenal karena membuat dokumen-dokumen palsu era Nazi.

Digital forensics dan teknologi ilmiah lainnya berperan penting dalam pembuktian pemalsuan tersebut. Melalui analisis laboratorium forensik, ditemukan bahwa tinta yang digunakan dalam buku harian itu baru diproduksi setelah 1945, yaitu setelah kematian Hitler. Selain itu, bahan kertas dan lem yang digunakan juga tidak sesuai dengan bahan yang umum digunakan pada masa Nazi. Pemeriksaan kimia menggunakan spektrometri dan teknik kromatografi menunjukkan adanya bahan modern yang tidak mungkin ditemukan pada dokumen asli dari tahun 1930-1940-an. Hal ini memperkuat bahwa buku harian itu adalah hasil rekayasa kontemporer.

Selain teknik kimia, analisis forensik digital juga dilakukan terhadap tulisan tangan dan tanda tangan yang ada dalam buku harian tersebut. Dengan memindai dan membandingkan pola tulisan dengan dokumen Hitler yang telah diverifikasi keasliannya, ditemukan banyak inkonsistensi gaya, tekanan pena, dan struktur huruf. Teknologi digital handwriting analysis membantu membuktikan bahwa tulisan tersebut tidak ditulis oleh tangan Hitler, melainkan oleh Kujau yang menirukan gayanya. Analisis lebih lanjut juga menunjukkan bahwa beberapa bagian teks merupakan kutipan dari buku atau dokumen publik yang sudah beredar, menunjukkan bahwa isi buku harian itu direka ulang berdasarkan sumber-sumber yang mudah diakses.

Akhirnya, Konrad Kujau dan wartawan Stern yang terlibat dalam penjualan dokumen itu diadili dan dijatuhi hukuman penjara. Skandal ini tidak hanya mempermalukan majalah Stern tetapi juga menyoroti pentingnya verifikasi forensik dalam dunia jurnalistik dan sejarah. Kasus Hitler's Diaries menjadi contoh monumental tentang bagaimana kombinasi forensik tradisional dan digital dapat membongkar kebohongan besar dan menyelamatkan catatan sejarah dari manipulasi. Sejak itu, pendekatan digital forensics menjadi semakin penting dalam mengautentikasi dokumen, baik analog maupun digital, dalam berbagai kasus hukum dan sejarah.

Pemalsuan Hitler's Diaries dibongkar bukan hanya oleh satu teknik, melainkan oleh kombinasi digital forensics, kimia forensik, dan sejarah

digital. Setiap teknik memberikan lapisan bukti yang saling mendukung, hingga akhirnya membuktikan bahwa dokumen tersebut adalah buatan tangan Konrad Kujau di era modern. Kasus ini menjadi pelajaran monumental bagi dunia jurnalistik, sejarah, dan kriminologi bahwa tidak ada dokumen — betapapun otentiknya kelihatannya — yang tak dapat diuji kebenarannya secara ilmiah dan digital.

a. Analisis Tinta (Ink Analysis)

Teknik ini menggunakan metode kimia seperti spektrometri massa, spektroskopi inframerah (FTIR), dan kromatografi lapis tipis (TLC) untuk menganalisis komposisi tinta. Dalam kasus Hitler's Diaries, ditemukan bahwa tinta yang digunakan mengandung senyawa kimia modern yang baru dikembangkan setelah tahun 1945, terutama jenis resin sintetis dan pelarut yang tidak ada pada zaman Nazi. Fakta ini langsung membantah klaim bahwa dokumen tersebut ditulis oleh Hitler semasa hidupnya. Ini merupakan bukti kimia yang kuat tentang ketidakotentikan dokumen.

b. Analisis Kertas (Paper Forensics)

Tim forensik memeriksa jenis, serat, dan kandungan kimia kertas. Mereka menemukan bahwa kertas yang digunakan dalam buku harian tersebut memiliki bahan tambahan pemutih optik (optical brighteners) yang tidak digunakan dalam pembuatan kertas sebelum tahun 1950-an. Selain itu, melalui mikroskop elektron dan uji spektroskopi sinar-X, diketahui bahwa struktur serat selulosa dan proses pemrosesan pulp kertas tersebut tidak cocok dengan standar produksi kertas Jerman era 1930-an atau 1940-an. Ini menunjukkan bahwa dokumen tersebut adalah produk pascaperang.

c. Forensik Tulisan Tangan (Handwriting Forensics & Digital Handwriting Analysis)

Tulisan tangan dalam buku harian dianalisis dan dibandingkan secara digital dengan tulisan Hitler yang telah diverifikasi. Software digitasi tulisan tangan digunakan untuk menganalisis bentuk huruf, tekanan goresan, sudut kemiringan, serta pola karakteristik khas dari tulisan asli Hitler. Para pakar menemukan bahwa tulisan dalam buku harian tidak

konsisten dengan gaya Hitler yang sesungguhnya, dan bahkan cenderung memiliki variasi kasar yang menandakan peniruan atau pemalsuan. Teknologi comparative handwriting analysis sangat membantu membuktikan bahwa Kujau-lah yang menulis semuanya dengan meniru gaya Hitler.

d. Analisis Cetakan dan Binding (Print and Binding Forensics)

Dokumen tersebut memiliki jilid dan lem yang diperiksa menggunakan mikroskop dan pengujian bahan kimia. Lem perekat yang digunakan dalam penjilidan ternyata adalah jenis synthetic glue modern, yang baru populer di era 1970-an dan 1980-an. Selain itu, terdapat bekas cetakan mesin modern yang tidak cocok dengan teknik penjilidan dan pencetakan yang digunakan oleh pemerintah Nazi pada waktu itu. Dengan demikian, dokumen ini dibuktikan tidak mungkin berasal dari era Hitler.

e. Image Forensics (Forensik Citra Digital)

Setelah dokumen dipindai, analisis noise visual dan struktur citra JPEG dilakukan untuk mencari perbedaan antara bagian teks yang berbeda. Misalnya, jika sebagian teks menunjukkan karakteristik kompresi atau distribusi noise yang tidak alami, ini bisa menunjukkan penyisipan teks atau gambar tambahan. Dalam kasus Hitler's Diaries, analisis semacam ini membantu memperkuat bahwa visualisasi dokumen tidak memiliki konsistensi alami dari dokumen asli yang ditulis tangan.

f. Spektroskopi Multispektral dan Ultraviolet (UV Light & Multi-spectral Imaging)

Teknik ini memungkinkan penyidik melihat detail tersembunyi yang tidak terlihat dalam cahaya biasa. Dengan menyinari dokumen menggunakan cahaya ultraviolet atau inframerah, terlihat perbedaan tingkat penuaan tinta dan kertas. Dalam kasus ini, tinta terlihat "baru" dibandingkan dengan kertasnya, yang menunjukkan bahwa tinta ditulis jauh setelah kertas tersebut dibuat (jika memang kertas itu tua). Ini adalah teknik visual dan ilmiah yang sangat kuat untuk mengungkap ketidaksesuaian waktu antara elemen dokumen.

g. Analisis Plagiarisme Isi (Content Forensics)

Beberapa kalimat dan paragraf dalam buku harian ternyata merupakan salinan dari sumber-sumber publik, termasuk pidato Hitler dan buku sejarah populer. Dengan menggunakan alat pendeteksi plagiarisme berbasis teks digital, penyidik dapat menelusuri bagian-bagian teks yang identik dengan dokumen-dokumen yang telah beredar sebelumnya. Ini menunjukkan bahwa isi buku harian tersebut bukanlah catatan pribadi, melainkan kompilasi kutipan yang disusun ulang secara manual oleh pemalsu.

h. Digital Metadata & Document History Analysis

Meskipun dokumen asli bersifat analog, salinan digital dokumen yang dikirim untuk publikasi dianalisis untuk metadata-nya. Metadata digital dari pemindai dan sistem komputer pengedit menunjukkan jejak waktu dan perangkat yang digunakan untuk memproses dokumen. Hal ini mencurigakan ketika ditemukan bahwa pemrosesan digital terjadi dalam waktu yang terlalu dekat dengan tanggal publikasi dan tidak mendukung narasi bahwa dokumen tersebut telah "disimpan rahasia selama puluhan tahun."

i. Analisis Sejarah Kontekstual Digital (Historical Timeline Cross-Verification)

Dengan menggunakan basis data digital sejarah, para penyidik memeriksa apakah catatan harian tersebut menyebutkan peristiwa, tanggal, atau orang yang tidak sesuai dengan pengetahuan sejarah saat itu. Ditemukan bahwa beberapa entri dalam buku harian mengacu pada kejadian atau istilah yang belum ada pada waktu yang diklaim. Ini adalah bentuk analisis konteks digital menggunakan cross-validation dengan arsip historis online.

h. Teknik Pelestarian dan Pembuktian Digital (Chain of Custody Digital Evidence)

Setelah dicurigai palsu, setiap salinan digital dari buku harian dijaga ketat untuk menjaga integritas bukti. Tim forensik menerapkan hashing (MD5/SHA) untuk memastikan tidak ada perubahan dalam file digital

selama investigasi. Bukti digital ini disusun sebagai bagian dari laporan forensik dan disajikan di pengadilan. Ini menunjukkan bagaimana teknik forensik digital tidak hanya membantu analisis, tapi juga menjamin validitas bukti dalam proses hukum.

KASUS PEMALSUAN KILLIAN DOCUMENT



Kasus pemalsuan Killian documents adalah salah satu kontroversi media terbesar di Amerika Serikat yang terjadi pada tahun 2004, menjelang pemilihan presiden AS. Dokumen-dokumen ini — yang disebut sebagai Killian documents — diklaim sebagai catatan pribadi Letnan Kolonel Jerry B. Killian, atasan George W. Bush saat Bush bertugas di Texas Air National Guard pada tahun 1970-an. Dokumen itu berisi tuduhan bahwa Bush tidak memenuhi kewajiban militernya dan menerima perlakuan istimewa untuk menghindari tugas aktif. Dokumen-

dokumen tersebut disiarkan oleh jurnalis CBS Dan Rather dalam acara 60 Minutes II, yang kemudian menimbulkan badai politik dan media.

Tak lama setelah siaran, keraguan teknik langsung mencuat, terutama dari kalangan blogger, jurnalis teknik, dan pakar teknologi. Mereka menunjukkan bahwa dokumen tersebut tampaknya menggunakan font Times New Roman, memiliki rata kanan (justified alignment), dan karakteristik typography seperti superscript (misalnya “th” dalam “111th”) — sesuatu yang sangat tidak umum, atau bahkan mustahil, dilakukan oleh mesin ketik biasa pada tahun 1970-an. Jenis tampilan ini lebih cocok dengan hasil cetakan Microsoft Word atau pengolah kata modern, bukan mesin ketik yang digunakan pada masa itu.

Para peneliti kemudian melakukan digital forensic typography analysis, yaitu membandingkan isi dokumen tersebut dengan hasil ketikan menggunakan Word Processor modern. Hasilnya mencengangkan: ketika dokumen diketik ulang menggunakan Microsoft Word versi 2004 dengan pengaturan default (font Times New Roman, ukuran 12, margin default) dan dibandingkan dengan salinan asli Killian documents, hasilnya 10eknik 10eknik10 secara pixel-by-pixel, termasuk penempatan huruf, jarak antar karakter, dan pemakaian superscript. Ini mengindikasikan bahwa dokumen kemungkinan besar dibuat menggunakan 10eknik1010 modern, bukan mesin ketik tahun 1970-an.

Lebih lanjut, tidak ada metadata atau jejak historis yang mendukung otentisitas dokumen tersebut. Dokumen itu muncul secara 10eknik dan diserahkan ke CBS oleh seorang perantara bernama Bill Burkett, seorang pensiunan perwira Garda Nasional yang merupakan pengkritik Bush. Burkett kemudian mengakui bahwa ia telah “memperoleh” dokumen dari sumber yang tidak jelas dan tidak dapat memverifikasi keasliannya. Tanpa rantai bukti (chain of custody) yang jelas, dokumen ini tidak dapat dianggap valid dalam penyelidikan forensik modern.

CBS dan Dan Rather menerima kecaman luas setelah investigasi menunjukkan kurangnya verifikasi forensik dan editorial atas dokumen tersebut sebelum ditayangkan. Komite 10eknik1010ent yang dibentuk oleh CBS mengkritik kelalaian dalam memeriksa keaslian dokumen dan kurangnya pengecekan fakta. Akibat dari kontroversi ini, empat staf CBS mengundurkan diri atau dipecat, dan reputasi Dan Rather sebagai jurnalis senior ikut tercoreng. Ia akhirnya mengundurkan diri dari CBS News tak lama setelahnya, meskipun tetap mempertahankan keyakinannya bahwa inti laporan itu benar, terlepas dari keaslian dokumennya.

Kasus Killian documents menjadi contoh penting bagaimana digital forensics, khususnya forensic typography dan document comparison, dapat digunakan untuk mengungkap pemalsuan dokumen dalam konteks politik dan media. Skandal ini juga mengajarkan pentingnya verifikasi digital dan kritisisme media, terutama di era digital ketika manipulasi

informasi bisa dilakukan dengan sangat meyakinkan. Ini adalah pengingat bahwa dokumen digital harus diperiksa tidak hanya dari isi, tapi juga dari struktur teknis, tampilan visual, dan konteks historis pembuatannya.

Setiap teknik digital forensics yang digunakan dalam kasus Killian Documents menunjukkan bahwa dokumen tersebut tidak mungkin berasal dari era 1970-an, melainkan dibuat menggunakan pengolah kata modern seperti Microsoft Word. Teknik-teknik seperti analisis font, superscript, kerning, perataan teks, dan overlay digital pixel membuktikan adanya penciptaan dokumen secara digital.

Kasus ini menegaskan pentingnya pengetahuan forensik digital tipografi dan metadata dalam menilai keaslian dokumen — terutama di era informasi saat pemalsuan bisa tampak meyakinkan di permukaan, namun terungkap dengan analisis teknis yang cermat.

a. Analisis Tipografi Digital (Digital Typography Forensics)

Teknik ini merupakan teknik awal dalam menilai keaslian visual dari dokumen Killian. Para ahli tipografi memeriksa font yang digunakan dalam dokumen tersebut dan menemukan bahwa font tersebut adalah Times New Roman, jenis huruf proporsional yang umum digunakan dalam Microsoft Word, bukan mesin ketik tahun 1970-an. Pada masa itu, mesin ketik militer hanya mendukung huruf monospasi seperti Courier. Penemuan ini sangat penting karena penggunaan font Times New Roman mengindikasikan bahwa dokumen kemungkinan besar diketik menggunakan pengolah kata modern, bukan dengan alat tulis era 1970-an.

b. Replikasi dengan Microsoft Word (Document Recreation Technique)

Untuk membuktikan kecurigaan tersebut, penyidik mengetik ulang isi dokumen menggunakan Microsoft Word 2004, dengan pengaturan default: Times New Roman 12 pt, margin normal, dan format standar. Hasilnya menunjukkan bahwa dokumen baru tersebut nyaris identik secara tata letak, panjang baris, dan struktur visual dengan dokumen Killian. Teknik ini digunakan untuk menunjukkan bahwa seluruh isi dokumen dapat direproduksi dengan

sempurna menggunakan perangkat lunak modern, sesuatu yang sangat tidak mungkin dilakukan oleh mesin ketik manual.

c. Perbandingan Piksel Digital (Pixel-by-Pixel Comparison)

Setelah dokumen asli dipindai menjadi gambar digital, para ahli melakukan analisis overlay dengan hasil Word yang diketik ulang. Gambar dokumen Word ditumpuk secara presisi di atas dokumen Killian dan dilakukan pengurangan citra (image subtraction) untuk melihat perbedaan posisi huruf dan jarak. Hasilnya membuktikan bahwa kedua dokumen nyaris 12eknik12 pada tingkat piksel, mengindikasikan bahwa bentuk dan tata letak dokumen sangat mungkin dihasilkan oleh 12eknik1212, bukan oleh mesin ketik tradisional yang selalu memiliki penyimpangan mekanik.

d. Analisis Superscript (Superscript Analysis)

Salah satu fitur yang sangat mencurigakan adalah penggunaan superscript otomatis, misalnya dalam “111th”. Mesin ketik tidak bisa menghasilkan superscript secara otomatis. Untuk melakukannya, operator harus menggeser kertas dan mengatur ulang posisi pengetikan — tindakan yang sangat rumit dan jarang dilakukan dalam dokumen formal militer. Sementara itu, Microsoft Word dapat menghasilkan superscript secara otomatis sebagai bagian dari autoformatting. Kehadiran superscript yang sempurna di dokumen Killian merupakan bukti digital yang sangat kuat bahwa dokumen ini berasal dari perangkat lunak modern.

e. Analisis Kerning (Kerning Forensics)

Kerning adalah 12eknik penyusunan teks di mana jarak antar huruf disesuaikan agar lebih estetik. Mesin ketik manual tidak memiliki kemampuan ini karena menggunakan spasi tetap (monospasi), sehingga setiap huruf memiliki lebar yang sama. Namun, dokumen Killian menunjukkan adanya kerning proporsional, misalnya pasangan huruf seperti “T” dan “o” yang lebih rapat 12eknik1212en “T” dan “a”. Dengan menggunakan perangkat lunak analisis gambar, para ahli membuktikan bahwa jarak antar huruf pada dokumen Killian bervariasi secara

konsisten, mengindikasikan bahwa dokumen dibuat dengan pengolah kata, bukan mesin ketik.

f. Analisis Perataan Paragraf (Justification Analysis)

Perataan teks kanan-kiri (justified text) adalah fitur lain yang mencurigakan. Mesin ketik tradisional hanya bisa menghasilkan teks dengan perataan kiri (ragged right), kecuali dengan intervensi manual. Sebaliknya, dokumen Killian menunjukkan teks yang sejajar sempurna di kedua sisi, sesuatu yang hanya bisa dicapai secara otomatis oleh perangkat lunak seperti Word. Dengan bantuan pengukuran visual digital, para ahli membuktikan bahwa batas kanan semua 13eknik1313e rata secara sempurna — menunjukkan bahwa dokumen ini diformat menggunakan sistem perataan digital, bukan ditulis secara manual.

g. Analisis Font Historis dan Timeline Teknologi

Untuk melengkapi analisis font, dilakukan penelusuran sejarah ketersediaan font Times New Roman dan teknologi pencetak huruf pada masa 1970-an. Hasilnya menunjukkan bahwa Times New Roman baru tersedia secara luas dalam pengolah kata 13eknik1313 pada akhir 1980-an dan menjadi default di Word pada tahun 1990-an. Selain itu, tidak ada mesin ketik militer pada 1973 yang menggunakan font tersebut. Ini adalah 13eknik forensik berbasis sejarah teknologi, yang membandingkan ketersediaan font dengan waktu pembuatan dokumen.

h. Pemeriksaan Metadata dan Asal Usul Dokumen (Provenance & Metadata Analysis)

Meskipun dokumen yang disiarkan CBS adalah hasil cetakan, para penyelidik tetap mencari jejak digital berupa metadata atau informasi sumber file asli. Sayangnya, tidak ada file Microsoft Word yang disediakan, tidak ada informasi dari printer, dan tidak ada penjelasan memadai dari sumber dokumen, Bill Burkett. Dalam dunia forensik digital, tidak adanya metadata dan rantai bukti (chain of custody) adalah 13eknik1313e kuat bahwa dokumen tidak bisa diverifikasi dan sangat rentan terhadap manipulasi.

i. Analisis Konteks Sejarah dan Kesesuaian Teknologi (Technological Contextual Analysis)

Teknik ini mengkaji apakah fitur-fitur dalam dokumen sesuai dengan teknologi yang tersedia pada waktu itu. Para ahli membandingkan mesin ketik yang digunakan di pangkalan militer dengan fitur-fitur seperti font, jarak spasi, superscript, dan alignment. Hasilnya menunjukkan bahwa tidak ada mesin ketik pada tahun 1973 yang mampu mereproduksi dokumen dengan fitur seperti yang dimiliki oleh Killian documents. Ini menjadi bukti tambahan bahwa dokumen tersebut berasal dari era digital.

j. Verifikasi Mandiri oleh Komunitas Teknologi (Open Source Peer Review)

Berbeda dengan investigasi internal, dalam kasus ini sebagian besar temuan forensik 14 teknik dari komunitas blogger, desainer, dan analis tipografi 14 teknik 14 ent. Mereka menggunakan 14 teknik terbuka dan mereplikasi eksperimen secara mandiri. Keberhasilan mereka menunjukkan bahwa forensik digital tidak hanya bergantung pada 14 teknik 14 formal, tetapi juga bisa dilakukan oleh komunitas yang memiliki akses ke perangkat lunak, logika analitis, dan pengetahuan teknis. Teknik ini secara tidak langsung memperkuat validitas bukti dengan pendekatan peer-reviewed terbuka.

Kesepuluh 14 teknik di atas bekerja secara sinergis dalam membongkar pemalsuan dokumen Killian. Setiap 14 teknik memberikan bukti teknis dan visual bahwa dokumen tersebut adalah hasil produksi modern, bukan dokumen asli dari tahun 1973. Pendekatan forensik digital yang digunakan dalam kasus ini menjadi studi penting dalam dunia jurnalisme, politik, dan keamanan informasi.

POLEMIK PROSES AKADEMIK JOKO WIDODO

TRANSKRIP AKADEMIK TANPA TANDA TANGAN OTORITAS UGM



tangan Dekan Fakultas Kehutanan UGM tahun 1985 menjadi fokus utama yang meragukan keasliannya.

Transkrip nilai akademik merupakan dokumen resmi yang merekam riwayat pendidikan seseorang di institusi pendidikan tinggi dan hanya diakui jika memuat tanda tangan dan cap resmi dari otoritas yang berwenang, seperti Dekan fakultas. Dalam kasus transkrip akademik Joko Widodo, ketidakadaan tanda

Dekan memiliki wewenang formal untuk mengesahkan transkrip akademik, dan tanda tangannya menjadi bukti sah bahwa dokumen tersebut telah diverifikasi dan diakui secara resmi oleh institusi. Ketidadaan tanda tangan ini secara langsung menghilangkan validitas dokumen sebagai dokumen resmi.

Prosedur pengesahan transkrip akademik di UGM pada tahun 1985 mewajibkan setiap dokumen akademik memuat tanda tangan Dekan serta stempel resmi fakultas. Hal ini menjadi standar penting untuk menjaga integritas dan keaslian dokumen tersebut.

Analisa visual yang dilakukan oleh Dirtipidum menunjukkan bahwa transkrip yang dipertanyakan tidak memuat tanda tangan Dekan yang asli. Ada indikasi bahwa dokumen tersebut merupakan salinan cetak atau fotokopi yang tidak mendapatkan pengesahan manual dari otoritas fakultas.

Lebih jauh, ditemukan ketidaksesuaian format dan tata letak dokumen dengan standar transkrip resmi UGM pada masa itu, terutama pada bagian tanda tangan yang seharusnya memuat tanda tangan Dekan, namun justru kosong atau tidak konsisten.

Cap resmi fakultas juga menjadi elemen penting untuk validasi dokumen. Namun pada transkrip ini, cap yang ada terlihat samar dan tidak sesuai dengan cap resmi Fakultas Kehutanan UGM tahun 1985, yang menambah keraguan atas keaslian dokumen.

Perbandingan dengan transkrip resmi lain dari UGM pada tahun 1980-an memperlihatkan perbedaan mencolok, khususnya dalam hal penempatan tanda tangan dan cap, sehingga menegaskan bahwa dokumen yang dipertanyakan bukan transkrip asli yang diterbitkan fakultas.

Ketidadaan tanda tangan Dekan dan sejumlah kejanggalan format dokumen menimbulkan dugaan kuat bahwa transkrip tersebut adalah dokumen palsu atau hasil modifikasi untuk memberikan kesan asli. Pemalsuan dokumen akademik adalah tindakan ilegal yang sangat merusak reputasi institusi dan individu.

161 SKS

DAFTAR NILAI PROGRAM S₁ (Lanjutan)

39.	KTM	210	Ilmu Ukur Tanah	3	C
40.	KTM	211	Penafsiran Potret Udara	3	B
41.	KTM	321	Pemungutan Hasil Hutan	2	B
42.	KTM	330	Ekonomi Kehutanan	2	C
43.	KTM	340	Metode Inventarisasi Hutan	2	A
44.	KTM	341	Manajemen Hutan I	3	B
45.	KTM	342	Manajemen Hutan II	2	A
46.	KTM	403	Statistik II	3	C
47.	KTM	431	Politik Kehutanan	2	B
48.	KTM	443	Manajemen Hutan III	3	A
49.	KTT	200	Struktur dan Sifat Kayu I	3	C
50.	KTT	310	Pengolahan Hasil Hutan	2	B
51.	KTK	300	Hidrologi Hutan	2	A
52.	KTK	310	Azaz-asas Pengelolaan Suaka Alam	2	B
53.	KTU	190	Pengantar Ilmu Kehutanan	1	C
54.	KTU	299	Kuliah Lapangan	1	B
55.	KTU	499	Praktek Lapangan	4	B
56.	KTU	599	Problema Kehutanan	4	B
MATA PELAJARAN PILIHAN :					
57.	KT	403	Tataaksana Keuangan	2	B
58.	KTM	302	Teknik Pencuplikan	3	A
59.	KTB	220	Genetika Hutan	2	B
60.	KTB	330	Ilmu Tanah Hutan	2	C
61.	KTB	473	Ekologi Hutan II	3	C
62.	KTT	403	Kimia Kayu	3	B
63.	KTB	413	Silvikultur Tropika	2	C
64.	KTK	411	Konservasi Suaka Alam	3	B
65.	KTT	401	Struktur dan Sifat Kayu II	2	B
66.	KTK	442	Andal	2	B
67.	KTK	420	Pengeringan/ Pengaw. Kayu	2	B
68.					
69.					
70.					
71.					
72.					

Dekan

Prof. Dr. Soenardi Prawirohatmojo
N I P : 130 120 347

Penjabat Dekan I

Dr. Ir. Djoko Marsono
N I P : 130 367 353

Motif pemalsuan dokumen ini diduga untuk memperoleh keuntungan politik dan sosial dengan mengklaim latar belakang pendidikan yang sebenarnya tidak dimiliki. Hal ini menjadi persoalan serius yang harus ditangani secara hukum.

Lebih lanjut, analisa spesifik terkait perbedaan format penulisan nilai dalam transkrip akademik Fakultas Kehutanan UGM tahun 1980-an dengan dokumen yang ditunjukkan oleh Dirtipidum Brigjend Djuhandhani Rahardjo Puro pada konferensi pers 22 Mei 2025 mengungkapkan kegagalan yang sangat signifikan.

Secara historis, transkrip akademik resmi Fakultas Kehutanan UGM pada masa 1980-an memiliki format yang relatif seragam dan terstandarisasi. Salah satu ciri pentingnya adalah seluruh nilai mata kuliah dicetak secara manual dengan mesin ketik atau diketik secara rapi, bukan ditulis tangan. Ini sesuai dengan prosedur administrasi di fakultas dan universitas pada masa itu, di mana pengisian transkrip dilakukan oleh staf administrasi menggunakan mesin ketik untuk menjaga konsistensi, keterbacaan, dan kejelasan dokumen.

Penggunaan mesin ketik memastikan bahwa setiap transkrip memiliki tampilan yang formal dan profesional, menghindari risiko kesalahan baca akibat tulisan tangan yang bisa bervariasi dari satu dokumen ke dokumen lain. Hal ini juga memudahkan proses arsip dan verifikasi dokumen.

Sementara itu, dokumen transkrip akademik yang dipresentasikan oleh Dirtipidum pada tanggal 22 Mei 2025 menunjukkan adanya penulisan nilai yang tampak ditulis tangan, bukan diketik. Hal ini jelas bertentangan dengan prosedur resmi yang berlaku di Fakultas Kehutanan UGM pada tahun 1980-an.

Tulisan tangan pada dokumen akademik resmi pada masa itu bukanlah praktek yang umum dan tidak lazim untuk nilai mata kuliah dalam transkrip resmi, karena akan menimbulkan ketidakseragaman, keraguan terhadap keaslian, serta potensi pemalsuan dokumen.

Keberadaan nilai yang ditulis tangan pada dokumen yang dipertanyakan ini mengindikasikan bahwa dokumen tersebut bukan merupakan hasil keluaran resmi fakultas, melainkan kemungkinan merupakan dokumen yang dibuat atau dimanipulasi di luar prosedur administrasi yang sah.

Faktor ini merupakan salah satu bukti forensik kuat yang mendukung dugaan bahwa transkrip tersebut tidak sah dan bukan dokumen resmi yang diterbitkan oleh Fakultas Kehutanan UGM.

Ketiadaan standar penulisan nilai, yakni tidak menggunakan mesin ketik melainkan tulisan tangan, menunjukkan adanya pelanggaran prosedur administrasi yang dapat menimbulkan keraguan besar atas keabsahan dokumen.

Dalam konteks ini, analisa spesifik terhadap gaya penulisan nilai menjadi alat penting untuk mengidentifikasi keaslian dokumen akademik, terutama untuk dokumen yang berasal dari masa sebelum digitalisasi arsip.

Kesimpulannya, fakta bahwa nilai pada transkrip akademik yang dipertanyakan ditulis tangan, sementara standar fakultas saat itu adalah pengetikan manual dengan mesin ketik, memperkuat indikasi bahwa dokumen tersebut bukanlah transkrip resmi dan tidak sah sebagai bukti riwayat akademik yang valid.

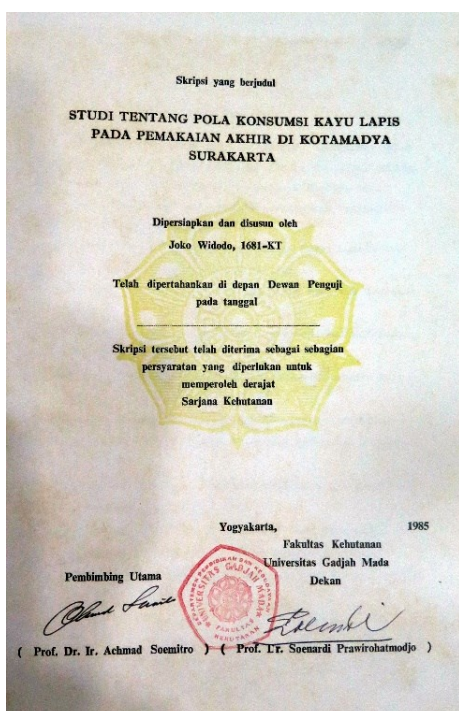
SKRIPSI TANPA TANDA TANGAN PENGUJI

Kasus skripsi Joko Widodo yang tidak memiliki lembar pengesahan penguji skripsi merupakan hal yang sangat krusial untuk dianalisa secara mendalam, mengingat lembar pengesahan penguji adalah salah satu syarat mutlak di UGM sebagai bukti bahwa sebuah skripsi telah melewati proses pengujian dan evaluasi akademik. Lembar pengesahan ini biasanya memuat tanda tangan dari penguji skripsi yang menjadi bukti otentik bahwa karya ilmiah tersebut telah diuji secara formal dan memenuhi standar akademik yang berlaku.

Tidak adanya lembar pengesahan penguji skripsi berarti bahwa proses verifikasi dan validasi terhadap skripsi tersebut tidak dilakukan atau tidak terekam secara resmi. Hal ini menimbulkan pertanyaan besar tentang keabsahan dan legalitas skripsi tersebut sebagai karya akademik yang sah. Di UGM, aturan sangat jelas bahwa skripsi harus mendapatkan

persetujuan dari penguji untuk dinyatakan layak sebagai salah satu syarat kelulusan sarjana.

Ketiadaan lembar pengesahan penguji juga mencerminkan adanya pelanggaran prosedur administrasi yang berlaku. Prosedur ini dirancang untuk menjamin kualitas akademik serta mencegah terjadinya plagiarisme dan pemalsuan. Bila skripsi tidak melalui proses pengesahan ini, maka tidak ada jaminan bahwa karya tersebut memenuhi standar akademik fakultas.



Lebih jauh, ketiadaan dokumen ini memperkuat indikasi bahwa skripsi tersebut mungkin tidak pernah diuji atau bahkan tidak pernah diselesaikan secara sah. Hal ini secara langsung berdampak pada keabsahan gelar sarjana yang diperoleh, karena gelar tersebut diberikan berdasarkan penyelesaian program akademik yang memenuhi

seluruh persyaratan, termasuk skripsi yang disahkan.

Jika skripsi tidak memiliki lembar pengesahan penguji, maka terdapat kemungkinan besar bahwa gelar sarjana kehutanan yang diklaim tidak didasarkan pada pemenuhan seluruh persyaratan akademik yang sah. Dengan kata lain, gelar tersebut bisa dianggap tidak valid atau bahkan palsu secara administratif.

Ketidaksesuaian ini bukan hanya soal kelengkapan dokumen, tetapi menyangkut integritas akademik secara keseluruhan. Gelar sarjana adalah bukti resmi kompetensi dan pencapaian akademik, yang harus didukung oleh bukti fisik berupa dokumen pengesahan skripsi yang sah.

Lebih lanjut, ketidakhadiran lembar pengesahan ini menimbulkan keraguan bahwa ada upaya manipulasi dokumen atau penghilangan dokumen penting yang seharusnya menjadi bagian dari arsip akademik resmi fakultas.

Dalam konteks penyidikan forensik dokumen, ketiadaan lembar pengesahan skripsi ini merupakan salah satu bukti kuat yang digunakan untuk mendukung dugaan bahwa terdapat pemalsuan dokumen akademik yang lebih luas, termasuk di antaranya transkrip nilai dan dokumen pendukung lainnya.

Aturan di UGM mengharuskan seluruh proses akademik, termasuk pengesahan skripsi, terdokumentasi dengan baik dan dapat diverifikasi. Tidak terpenuhinya hal ini menandakan bahwa prosedur akademik telah dilanggar atau diabaikan, sehingga keabsahan gelar sarjana menjadi dipertanyakan.

Kepalsuan skripsi ini juga berdampak pada kredibilitas Fakultas Kehutanan UGM sebagai institusi pendidikan yang seharusnya menjaga kualitas akademik dan integritas lulusan. Jika ada lulusan yang mendapatkan gelar tanpa memenuhi prosedur akademik yang sah, maka reputasi fakultas dan universitas bisa terancam.

Fenomena ini juga dapat memicu keresahan di kalangan mahasiswa dan alumni lain yang menuntut kesetaraan dan keadilan dalam proses akademik. Mereka berhak mendapatkan kepastian bahwa gelar yang diterima benar-benar mencerminkan usaha dan pencapaian akademik yang valid.

Lebih luas, kasus ini bisa menjadi preseden buruk jika tidak ditangani dengan transparan dan tegas. Hal ini berpotensi mendorong praktik serupa di masa depan, yang merusak tatanan pendidikan tinggi di Indonesia.

Pihak universitas sebagai institusi pendidikan wajib melakukan audit internal dan investigasi menyeluruh untuk memastikan keabsahan

seluruh dokumen akademik yang terkait dan mengungkap setiap indikasi pelanggaran.

Jika ditemukan ketidaksesuaian atau kepalsuan, universitas juga harus berani mengambil tindakan tegas, termasuk pencabutan gelar jika memang terbukti tidak sah.

Transparansi dalam proses investigasi ini sangat penting agar publik mendapat kejelasan dan kepercayaan terhadap penanganan kasus.

Analisa ini menegaskan bahwa lembar pengesahan penguji skripsi bukan hanya sekadar dokumen administratif, tetapi bagian integral yang menentukan validitas seluruh proses akademik dan gelar yang diterbitkan.

Ketidakhadiran lembar pengesahan ini menjadi bukti yang sangat memberatkan dugaan kepalsuan dokumen akademik yang tidak dapat diabaikan.

Secara keseluruhan, kepalsuan skripsi tanpa lembar pengesahan penguji sangat berkaitan erat dengan dugaan kepalsuan gelar sarjana kehutanan UGM, yang secara substansial menghilangkan legalitas dan keabsahan akademik gelar tersebut.

Oleh karena itu, temuan ini menuntut penyelidikan lebih lanjut dan tindakan hukum agar prinsip kejujuran dan integritas akademik tetap terjaga demi masa depan pendidikan tinggi di Indonesia.

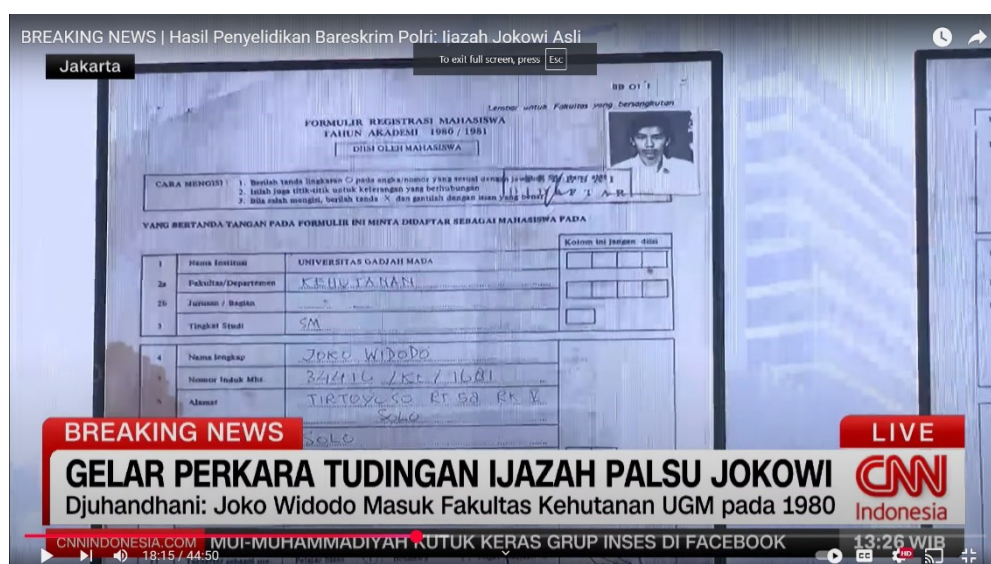
JOKO WIDODO TERDAFTAR PROGRAM SARJANA MUDA

Dalam sistem pendidikan tinggi Indonesia era 1980-an, status “sarjana muda” mengacu pada mahasiswa yang telah menyelesaikan pendidikan jenjang menengah perguruan tinggi, setara dengan diploma tiga (D-3) atau program setengah dari strata satu (S-1). Mahasiswa dengan status ini belum mendapatkan gelar “Sarjana” penuh karena masih harus

melanjutkan ke tahap lanjutan untuk menyelesaikan persyaratan akademik yang lebih tinggi.

Fakultas Kehutanan UGM pada periode tersebut menganut sistem berjenjang: mahasiswa lebih dulu menempuh program sarjana muda selama kira-kira tiga tahun, kemudian baru dapat melanjutkan ke program sarjana lengkap yang membutuhkan tambahan dua tahun atau lebih, tergantung penyelesaian skripsi.

Bila Jokowi tercatat sebagai mahasiswa sarjana muda hanya pada tahun akademik 1980/1981 dan 1981/1982, berarti ia baru menempuh atau menyelesaikan tahap awal dari pendidikan tinggi tersebut. Data ini menunjukkan bahwa pada dua tahun tersebut, ia belum masuk ke program sarjana penuh yang memerlukan waktu tambahan.



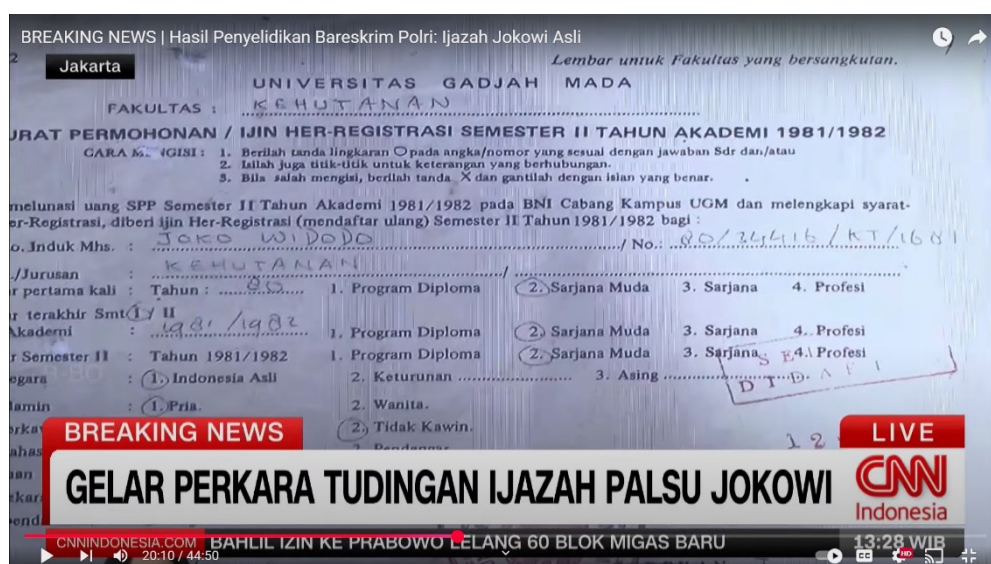
Untuk mendapatkan gelar sarjana kehutanan (S.Hut.) di UGM era itu, mahasiswa wajib menuntaskan program lanjutan setelah sarjana muda, termasuk menyusun skripsi dan mengikuti ujian pendadaran. Semua ini membutuhkan waktu tambahan paling sedikit dua tahun lagi setelah lulus sarjana muda.

Dengan kata lain, jika Jokowi hanya tercatat pada dua tahun akademik di tingkat sarjana muda dan tidak ada catatan resmi kelanjutannya di

tingkat sarjana penuh, maka secara administratif ia belum memenuhi syarat mendapatkan gelar sarjana kehutanan.

Secara logis, mahasiswa yang berhenti atau tidak melanjutkan setelah sarjana muda tidak akan otomatis menerima gelar sarjana. Sistem di UGM tidak mengenal mekanisme “loncat” dari sarjana muda langsung ke gelar sarjana tanpa menempuh program lanjutannya.

Bahkan bila diasumsikan ia menyelesaikan sarjana muda di akhir tahun akademik 1982/1983, maka ia harus tercatat minimal dua tahun lagi untuk menyelesaikan strata satu. Jika catatan registrasi mahasiswa tidak menunjukkan keberlanjutan pada tahun-tahun berikutnya, hal ini menjadi indikasi kuat bahwa proses akademik tersebut terhenti.



Ketiadaan bukti kelulusan dari program sarjana penuh memperkuat dugaan bahwa gelar yang disandang kemudian tidak diperoleh melalui jalur akademik yang semestinya. Apalagi, UGM memiliki sistem dokumentasi ketat terhadap daftar lulusan, sidang pendadaran, dan ijazah resmi yang diterbitkan.

Status “sarjana muda” sendiri pada masa itu sering dijadikan titik keluar mahasiswa yang ingin langsung bekerja atau melanjutkan studi di tempat

Konferensi pers Dirtipidum pada 22 Mei 2025 menjadi titik penting dalam analisis ini. Dalam presentasi tersebut, pihak kepolisian menampilkan dokumen akademik yang diklaim sebagai milik Jokowi. Dokumen itu mencantumkan total SKS yang telah diselesaikan — hanya 122 SKS. Secara objektif, angka ini berada jauh di bawah ambang minimal 161 SKS yang disyaratkan UGM untuk gelar Sarjana Kehutanan.

Jika kita menghitung jarak kekurangannya, $161 \text{ SKS} - 122 \text{ SKS} = 39 \text{ SKS}$ belum terpenuhi. Dalam sistem pendidikan tinggi, 39 SKS setara dengan lebih dari satu tahun akademik penuh, bahkan mendekati dua semester penuh dengan beban maksimum. Artinya, seseorang dengan total 122 SKS jelas belum memenuhi syarat untuk yudisium sarjana.

122 SKS umumnya merupakan akumulasi yang wajar bagi mahasiswa yang berhenti di tingkat Sarjana Muda. Hal ini menguatkan dugaan bahwa Jokowi tidak melanjutkan ke tahap Sarjana penuh. Tanpa menempuh tambahan 39 SKS yang mencakup mata kuliah lanjutan dan skripsi, tidak mungkin ia lulus sebagai Sarjana Kehutanan dari UGM secara prosedural.

Kondisi ini membuka kemungkinan besar bahwa proses studinya terhenti sebelum mencapai tahap akhir. Dalam terminologi kampus, ini setara dengan status “drop out” atau tidak melanjutkan ke jenjang berikutnya setelah menyelesaikan sarjana muda.

Dengan demikian, bila hanya berbekal catatan sarjana muda di dua tahun akademik, maka secara hukum akademik dan logika kurikulum, Jokowi tidak mungkin menjadi lulusan sarjana kehutanan UGM. Tidak ada celah regulasi pada era itu yang memungkinkan lompatan akademis semacam ini.

Karena itu, melihat seluruh rangkaian data, fakta, dan aturan yang berlaku pada masa tersebut, logika akademik menyimpulkan bahwa Jokowi tidak menyelesaikan studi sarjana di Fakultas Kehutanan UGM. Status terakhir yang tercatat hanyalah mahasiswa sarjana muda.

Dari perspektif akademik murni, hal ini berarti ia berhenti sebelum mencapai tingkat sarjana penuh — atau dalam bahasa sederhananya, ia adalah mahasiswa yang drop out dari program sarjana kehutanan UGM setelah tahap sarjana muda.

BANTAHAN KASMUDJO DALAM PERANNYA MEMBIMBING JOKOWI WIDODO

Kasmudjo dalam bantahannya menegaskan bahwa ia tidak pernah tercatat maupun diadministrasikan secara resmi sebagai dosen pembimbing akademik Jokowi, juga tidak pernah menjadi dosen pembimbing skripsi Jokowi. Pernyataan ini sangat penting karena dalam sistem akademik UGM, pembimbing skripsi dan pembimbing akademik memiliki catatan administratif yang jelas. Jika benar Jokowi adalah lulusan resmi, tentu status pembimbingan itu harus tercatat, apalagi bila klaimnya digunakan untuk menguatkan keabsahan ijazah dan skripsinya. Bantahan Kasmudjo meruntuhkan argumen yang mencoba membangun legitimasi akademik Jokowi dengan menghadirkan sosok dosen yang seolah membimbingnya.

Ketika bantahan Kasmudjo ini dikaitkan dengan fakta lain yang disampaikan Diritpidum, yakni jumlah SKS Jokowi yang hanya 122, masalahnya menjadi semakin serius. Di UGM, syarat kelulusan program sarjana adalah minimal 161 SKS. Dengan hanya 122 SKS, Jokowi tidak mungkin memenuhi standar akademik untuk meraih gelar sarjana. Artinya, tanpa tambahan bukti valid mengenai perpanjangan studi, konversi SKS, atau kebijakan khusus yang sah secara administratif, klaim kelulusan tersebut gugur dengan sendirinya.

Selain masalah SKS, ketiadaan lembar pengesahan penguji skripsi juga semakin memperkuat keraguan. Dalam tradisi akademik, lembar pengesahan penguji adalah dokumen mutlak yang menunjukkan bahwa mahasiswa telah melalui ujian skripsi, disidangkan, dan disahkan oleh dosen penguji. Jika dokumen ini tidak ada, maka secara hukum akademik skripsi itu tidak pernah diuji. Karena skripsi adalah syarat wajib

kelulusan sarjana, ketiadaan dokumen ini membuat klaim kelulusan menjadi tidak sah.

Bantahan Kasmudjo juga semakin bermakna bila dihubungkan dengan tidak adanya tanda tangan dekan dan pembantu dekan pada transkrip akademik yang ditunjukkan Dirtipidum. Transkrip yang sah dari sebuah fakultas harus disahkan dengan tanda tangan pejabat akademik berwenang, minimal dekan atau wakil dekan I. Tanpa pengesahan tersebut, transkrip hanya sekedar daftar nilai internal, bukan dokumen akademik resmi. Hal ini mengindikasikan bahwa dokumen yang ditampilkan tidak memiliki kekuatan hukum akademik dan tidak bisa digunakan sebagai bukti kelulusan.

Jika kita satukan seluruh fakta ini—bantahan Kasmudjo, SKS hanya 122, ketiadaan lembar pengesahan penguji skripsi, dan tidak adanya tanda tangan dekan maupun pembantu dekan—maka terlihat adanya pola inkonsistensi yang serius. Setiap elemen yang seharusnya menopang keabsahan kelulusan justru tidak ada. Bantahan Kasmudjo semakin mempertegas bahwa nama dan statusnya tidak boleh dijadikan legitimasi untuk sebuah proses akademik yang dokumennya sendiri cacat atau tidak lengkap.

Dengan demikian, bantahan Kasmudjo terhadap tidak adanya peran pembimbingan skripsi dan akademik terhadap Jokowi yang disampaikan kepada Rismon Sianipar tidak sekedar soal personal, melainkan sebuah penegasan akademik bahwa klaim pembimbingan itu palsu. Apalagi jika dikaitkan dengan dokumen bermasalah yang dipublikasikan Dirtipidum, maka jelas bahwa konstruksi narasi kelulusan Jokowi semakin runtuh.

POTENSI REKAYASA LEMBAR PENGESAHAN SKRIPSI

Pertanyaan mengenai lembar pengesahan skripsi yang tidak mencantumkan tanda tangan dosen pembimbing dan penguji, khususnya yang diklaim sebagai milik rekan-rekan mahasiswa Joko Widodo (Lembah Ediyanto, Edy Triyanto, Sigit Hardwinarto, dan Sri Dominingsih), menimbulkan sejumlah pertanyaan penting terkait

validitas dan keabsahan dokumen tersebut. Sebuah dokumen pengesahan yang tidak mencantumkan tanda tangan dari pihak yang berwenang — yaitu dosen pembimbing dan penguji — memiliki dampak yang serius terhadap sahnya sebuah skripsi secara akademik.



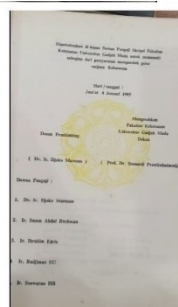
**Sri Dominingsih
Kehutanan
Lulus 1985**



**Sigit
Hardwinarto
Kehutanan
Lulus 1985**



**Edy Triyanto
Kehutanan
Lulus 1985**



**Lembah Ediyanto
Kehutanan Lulus
1984**



**Joko Widodo
Kehutanan
Lulus 1985**

Lembar pengesahan skripsi adalah dokumen utama dalam prosedur kelulusan mahasiswa di banyak perguruan tinggi. Dokumen ini berfungsi sebagai pengesahan bahwa skripsi yang diajukan oleh mahasiswa telah melalui tahap ujian pendadaran dan dinyatakan layak oleh dosen pembimbing dan penguji. Lembar pengesahan tanpa tanda tangan dari dosen pembimbing dan penguji menunjukkan bahwa skripsi tersebut tidak pernah melalui proses ujian formal yang diperlukan untuk menyelesaikan pendidikan.

Ketiadaan tanda tangan ini dapat menandakan adanya kejanggalan administratif atau bahkan manipulasi dokumen, yang dapat merugikan kredibilitas kampus dan integritas akademik. Hal ini menjadi lebih mencurigakan ketika dokumen tersebut diklaim sebagai milik mahasiswa yang terlibat dalam studi bersama Joko Widodo, dan seharusnya mencakup validasi oleh pihak berwenang.

Lembar pengesahan skripsi adalah dokumen utama dalam prosedur kelulusan mahasiswa di banyak perguruan tinggi. Dokumen ini berfungsi sebagai pengesahan bahwa skripsi yang diajukan oleh mahasiswa telah

melalui tahap ujian pendadaran dan dinyatakan layak oleh dosen pembimbing dan penguji. Lembar pengesahan tanpa tanda tangan dari dosen pembimbing dan penguji menunjukkan bahwa skripsi tersebut tidak pernah melalui proses ujian formal yang diperlukan untuk menyelesaikan pendidikan.

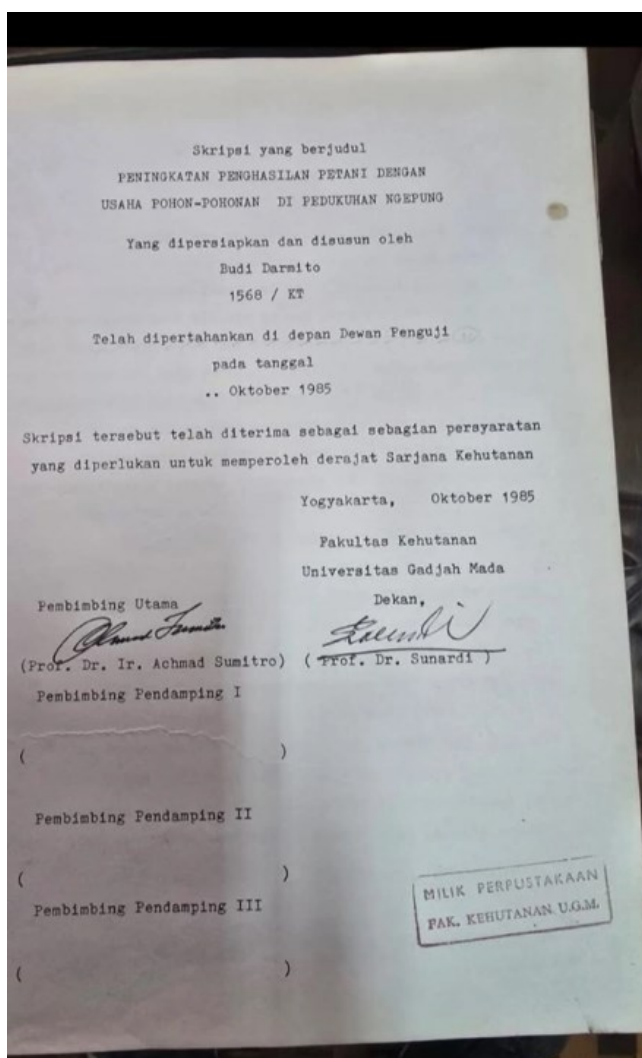
Ketiadaan tanda tangan ini dapat menandakan adanya kejanggalan administratif atau bahkan manipulasi dokumen, yang dapat merugikan kredibilitas kampus dan integritas akademik. Hal ini menjadi lebih mencurigakan ketika dokumen tersebut diklaim sebagai milik mahasiswa yang terlibat dalam studi bersama Joko Widodo, dan seharusnya mencakup validasi oleh pihak berwenang.

Kembali ke pertanyaan mengenai lembar pengesahan skripsi yang diklaim oleh Joko Widodo atau rekan-rekannya dan ditulis menggunakan font Times New Roman, hal ini menimbulkan ketidakcocokan historis dalam konteks waktu. Jika skripsi tersebut benar ditulis pada tahun 1985 atau lebih awal, maka penggunaan Times New Roman pada dokumen pengesahan tersebut bisa menjadi indikasi adanya manipulasi atau pemalsuan dokumen.

Pada tahun 1985, Times New Roman belum tersedia secara luas untuk publik, terutama di luar sistem komputer yang berbasis pada Microsoft Windows. Perangkat pengolah kata seperti Microsoft Word yang memuat font tersebut baru mulai tersedia pada awal 1990-an. Jika dokumen tersebut benar berasal dari 1985 atau sebelumnya, dan font Times New Roman digunakan, maka ini menjadi kesalahan historis yang mencurigakan, karena pada waktu itu font ini belum lazim digunakan.

Keberadaan font yang tidak sesuai dengan periode waktu tertentu pada dokumen penting seperti lembar pengesahan skripsi bisa menjadi petunjuk adanya pemalsuan informasi. Di dunia akademik, keabsahan dokumen sangat penting karena dokumen skripsi adalah representasi dari proses evaluasi akademik yang harus transparan dan dapat dipertanggungjawabkan. Apabila ada indikasi pemalsuan atau manipulasi dokumen, maka kepercayaan terhadap integritas proses akademik dan institusi pendidikan itu sendiri akan terganggu. Ini

mempengaruhi kredibilitas universitas dalam mempertahankan reputasinya sebagai lembaga pendidikan yang sah dan transparan.



Font Times New Roman, meskipun sangat umum digunakan saat ini, belum ada pada tahun 1985. Penggunaan font ini dalam dokumen yang diklaim milik mahasiswa pada waktu itu mengindikasikan adanya ketidaksesuaian historis yang sangat mencurigakan. Jika hal ini benar, maka manipulasi dokumen atau kesalahan pengarsipan yang serius bisa saja terjadi. Hal ini menggugah pertanyaan mendalam tentang keabsahan skripsi yang bersangkutan dan bisa menurunkan kredibilitas akademik baik dari segi individu maupun institusi.

Sebaliknya, skripsi atas nama Budi Darmito yang juga berasal dari tahun 1985 terlihat masih menggunakan mesin ketik manual, sebagaimana lazimnya pada masa tersebut. Ini menguatkan dugaan bahwa dokumen skripsi yang benar-benar ditulis dan disahkan pada tahun tersebut umumnya memiliki ciri khas ketikan mekanis dan bukan hasil word processor. Kontras ini memberikan indikasi bahwa adanya skripsi-

skripsi dengan font digital tanpa tanda tangan dan format standar bisa jadi bukan naskah asli atau telah dimodifikasi, sehingga menimbulkan dampak negatif berupa menurunnya kepercayaan publik terhadap dokumentasi sejarah akademik suatu institusi, apalagi bila menyangkut tokoh nasional.

UJI LINTASAN STEMPEL

ANALISA SEBARAN INTENSITAS RGB PADA FOTO IJAZAH JOKOWI

Program yang dirancang penulis sebuah aplikasi GUI (Graphical User Interface) yang dirancang menggunakan bahasa Python dengan pustaka Tkinter dan OpenCV. Tujuan utama dari program ini adalah untuk menganalisis sebaran intensitas warna RGB (Red, Green, Blue) pada area tertentu dari sebuah gambar, khususnya foto dokumen seperti ijazah Presiden Jokowi, guna menyelidiki keberadaan lintasan stempel. Melalui proses seleksi manual area gambar oleh pengguna, program ini memungkinkan eksplorasi detail dari distribusi warna di area yang diduga mengandung stempel basah, untuk mengetahui apakah cap tersebut benar-benar tercetak secara fisik atau justru hasil digitalisasi (editan).

Program bekerja dengan memungkinkan pengguna untuk membuka file gambar dari sistem komputer, seperti file ijazah dalam format PNG atau JPG. Setelah gambar dimuat, pengguna dapat menelusuri gambar secara visual dengan fitur scroll dan zoom, yang sangat membantu saat melakukan analisis dokumen beresolusi tinggi. Gambar ditampilkan dalam kanvas yang mendukung interaksi mouse, seperti zoom menggunakan roda scroll dan menggambar kotak seleksi menggunakan klik dan geser. Hal ini sangat penting karena pengguna harus dapat menentukan dengan akurat area stempel pada gambar tersebut untuk mendapatkan hasil analisis intensitas warna yang valid.

Setelah pengguna memilih area yang mencurigakan (misalnya letak stempel), program akan merekam koordinat seleksi dan membentuk

bounding box. Area dalam bounding box inilah yang disebut sebagai Region of Interest (ROI). ROI tersebut kemudian diambil dari gambar asli untuk dilakukan pemrosesan lebih lanjut. Gambar akan dibagi menjadi tiga saluran warna (channel) yakni R (merah), G (hijau), dan B (biru), sesuai dengan sistem warna digital RGB yang digunakan dalam representasi gambar.

Setiap channel warna tersebut akan dikonversi menjadi tabel numerik menggunakan pustaka `pandas`, yang kemudian ditampilkan dalam jendela baru menggunakan `pandastable`. Ini memungkinkan pengguna melihat nilai intensitas warna per piksel dalam area yang telah dipilih. Dengan mengamati nilai-nilai tersebut, seorang analis dapat memperkirakan apakah ada variasi intensitas yang wajar dan acak—ciri dari stempel asli yang ditekan secara fisik—atau sebaliknya, pola warna yang terlalu seragam dan halus, yang merupakan indikator kuat manipulasi digital.

Lebih lanjut, program ini tidak hanya menampilkan data intensitas dalam bentuk tabel, tetapi juga dalam bentuk histogram warna. Histogram memperlihatkan sebaran frekuensi intensitas warna (0–255) dalam bentuk grafik batang. Setiap channel warna (R, G, B) dibuatkan histogram terpisah, yang dapat dibandingkan untuk mengamati kecenderungan pola distribusi warna. Jika hasil histogram menunjukkan distribusi alami dengan puncak tidak simetris dan rentang variasi lebar, itu mengarah pada kemungkinan bahwa area tersebut mengandung tinta fisik asli. Namun jika histogramnya sangat simetris dan konsisten, itu dapat mengindikasikan bahwa gambar telah diedit atau stempel ditambahkan secara digital.

Dalam konteks menganalisis ijazah Presiden Jokowi, fitur ini memungkinkan analisis forensik digital non-destruktif terhadap area stempel pada dokumen. Stempel yang benar-benar dicap dengan tinta cenderung memiliki ketidakteraturan warna, karena tekanan tangan, distribusi tinta, dan interaksi dengan permukaan kertas tidak pernah seragam. Sebaliknya, stempel digital hasil salinan atau editan biasanya memiliki tekstur halus, warna terlalu konsisten, dan tepi terlalu bersih,

yang semuanya dapat dengan mudah ditangkap oleh distribusi data intensitas maupun histogram RGB.

Program ini sangat berguna karena memungkinkan pembuktian secara kuantitatif dan visual. Secara kuantitatif, nilai-nilai intensitas yang ditampilkan bisa diekspor atau dibandingkan. Secara visual, histogram dan tampilan gambar yang dapat diperbesar memberi wawasan tambahan tentang bagaimana distribusi warna terdistribusi secara spasial. Selain itu, pengguna bisa melakukan zoom untuk mengamati pixel demi pixel area stempel, mengidentifikasi apakah ada artefak digital seperti pixel berulang atau garis tepi yang tidak alami.

Teknik ini masuk dalam kategori digital image forensics, yaitu cabang dari ilmu forensik digital yang bertujuan untuk menganalisis keaslian gambar dengan cara memeriksa struktur internalnya. Meskipun tidak sekompleks teknik spektral atau deteksi pemalsuan dengan AI, pendekatan berbasis RGB intensity histogram ini cukup kuat sebagai bukti awal bahwa sebuah area telah dimanipulasi atau tidak. Bahkan dalam praktik investigasi jurnalistik atau forensik ringan, analisis histogram warna seperti ini sering digunakan untuk mendeteksi kejanggalan visual yang tidak bisa dilihat dengan mata telanjang.

Kesimpulannya, program ini dapat digunakan oleh peneliti, jurnalis, atau ahli forensik untuk menilai keaslian stempel pada dokumen digital seperti ijazah Presiden Jokowi, dengan menilai distribusi dan karakteristik warna di area yang diduga. Karena memberikan tampilan interaktif dan data numerik yang bisa ditelaah langsung, program ini merupakan alat bantu yang sangat bermanfaat dalam penyelidikan pemalsuan visual berbasis gambar.

SKRIP PROGRAM PYTHON ANALISA RUANG WARNA RGB

```
import tkinter as tk
from tkinter import filedialog, Toplevel
from PIL import Image, ImageTk
import cv2
import numpy as np
import pandas as pd
from pandastable import Table
import matplotlib.pyplot as plt
```

```

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

class ImageViewer:
    def __init__(self, root):
        self.root = root
        self.root.title("Image Viewer with Zoom and Intensity Bounding
Box")
        self.root.geometry("1200x800")

        self.original_image = None
        self.image = None
        self.tk_image = None
        self.bounding_box = None
        self.rect_id = None

        self.scale = 1.0 # Initial zoom level

        self.create_widgets()

    def create_widgets(self):
        canvas_frame = tk.Frame(self.root)
        canvas_frame.pack(fill=tk.BOTH, expand=True)

        self.canvas = tk.Canvas(canvas_frame, bg="gray")
        self.canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

        h_scroll = tk.Scrollbar(canvas_frame, orient=tk.HORIZONTAL,
command=self.canvas.xview)
        h_scroll.pack(side=tk.BOTTOM, fill=tk.X)

        v_scroll = tk.Scrollbar(canvas_frame, orient=tk.VERTICAL,
command=self.canvas.yview)
        v_scroll.pack(side=tk.RIGHT, fill=tk.Y)

        self.canvas.configure(xscrollcommand=h_scroll.set,
yscrollcommand=v_scroll.set)
        self.canvas.bind("<ButtonPress-1>", self.on_button_press)
        self.canvas.bind("<B1-Motion>", self.on_move_press)
        self.canvas.bind("<ButtonRelease-1>", self.on_button_release)

        # Mouse wheel zoom
        self.canvas.bind("<MouseWheel>", self.zoom) # Windows & Mac
        self.canvas.bind("<Button-4>", self.zoom) # Linux scroll up
        self.canvas.bind("<Button-5>", self.zoom) # Linux scroll down

        control_frame = tk.Frame(self.root)
        control_frame.pack(pady=10)

        self.btn_open = tk.Button(control_frame, text="Open Image",
command=self.open_image)
        self.btn_open.pack(side=tk.LEFT, padx=10)

    def open_image(self):
        file_path = filedialog.askopenfilename(filetypes=[("Image Files",
"*.png;*.jpg;*.jpeg;*.bmp")])
        if file_path:
            img = cv2.imread(file_path)

```

```

        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.original_image = img
        self.scale = 1.0
        self.display_image()

    def display_image(self):
        if self.original_image is not None:
            h, w = int(self.original_image.shape[0] * self.scale),
            int(self.original_image.shape[1] * self.scale)
            resized_image = cv2.resize(self.original_image, (w, h),
            interpolation=cv2.INTER_LINEAR)
            self.image = resized_image
            img_pil = Image.fromarray(resized_image)
            self.tk_image = ImageTk.PhotoImage(img_pil)
            self.canvas.delete("all")
            self.canvas.create_image(0, 0, anchor=tk.NW,
            image=self.tk_image)
            self.canvas.config(scrollregion=(0, 0, w, h))

    def zoom(self, event):
        if event.delta > 0 or event.num == 4:
            self.scale *= 1.1
        elif event.delta < 0 or event.num == 5:
            self.scale /= 1.1
        self.display_image()

    def on_button_press(self, event):
        self.start_x = self.canvas.canvasx(event.x)
        self.start_y = self.canvas.canvasy(event.y)
        if self.rect_id:
            self.canvas.delete(self.rect_id)
            self.rect_id = self.canvas.create_rectangle(self.start_x,
            self.start_y, self.start_x, self.start_y, outline='red', width=3)

    def on_move_press(self, event):
        cur_x = self.canvas.canvasx(event.x)
        cur_y = self.canvas.canvasy(event.y)
        self.canvas.coords(self.rect_id, self.start_x, self.start_y,
        cur_x, cur_y)

    def on_button_release(self, event):
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.bounding_box = (self.start_x, self.start_y, self.end_x,
        self.end_y)
        self.show_intensity()

    def show_intensity(self):
        if self.bounding_box and self.image is not None:
            x1, y1, x2, y2 = map(int, self.bounding_box)
            x1, y1 = max(0, x1), max(0, y1)
            x2, y2 = min(self.image.shape[1], x2),
            min(self.image.shape[0], y2)
            roi = self.image[y1:y2, x1:x2]
            if roi.size > 0:
                r, g, b = cv2.split(roi)
                df_r = pd.DataFrame(r)

```

```

        df_g = pd.DataFrame(g)
        df_b = pd.DataFrame(b)
        self.show_intensity_table(df_r, df_g, df_b)

    def show_intensity_table(self, df_r, df_g, df_b):
        new_window = Toplevel(self.root)
        new_window.title("Intensity Values")
        new_window.geometry("1900x700")

        frame_tables = tk.Frame(new_window)
        frame_tables.pack(fill=tk.BOTH, expand=True)

        # Create 3 columns for Red, Green, Blue
        for i, (df, color_name) in enumerate(zip([df_r, df_g, df_b],
["Red", "Green", "Blue"])):
            col_frame = tk.Frame(frame_tables)
            col_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

            label = tk.Label(col_frame, text=f"{color_name} Intensity")
            label.pack()

            table_frame = tk.Frame(col_frame)
            table_frame.pack(fill=tk.BOTH, expand=True)

            pt = Table(table_frame, dataframe=df, showtoolbar=True,
showstatusbar=True)
            pt.show()

            hist_frame = tk.Frame(col_frame)
            hist_frame.pack(fill=tk.BOTH, expand=True)

            self.plot_histogram(hist_frame, df.values.flatten(),
color_name[0].lower(), f"{color_name} Histogram")

    def plot_histogram(self, frame, data, color, title):
        fig, ax = plt.subplots(figsize=(6, 3))
        fig.tight_layout(pad=3.0)
        ax.hist(data, bins=256, color=color, alpha=0.7)
        ax.set_xlim([0, 256])
        ax.set_title(title)

        canvas = FigureCanvasTkAgg(fig, master=frame)
        canvas.draw()
        canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

if __name__ == "__main__":
    root = tk.Tk()
    app = ImageViewer(root)
    root.mainloop()

```

Program Python di atas adalah sebuah aplikasi GUI berbasis Tkinter yang dirancang untuk memfasilitasi analisis intensitas warna RGB pada gambar digital. Program ini sangat bermanfaat dalam konteks forensik digital dokumen, karena memungkinkan pengguna memeriksa area

tertentu dari gambar (misalnya ijazah atau dokumen resmi) untuk menganalisis karakteristik warna dan distribusinya, terutama untuk mendeteksi kemungkinan manipulasi digital seperti stempel palsu.

Dalam kelas `ImageViewer`, konstruktor `__init__` menetapkan elemen-elemen awal seperti jendela utama, variabel penyimpanan gambar, dan variabel pengontrol zoom serta bounding box. Ukuran jendela GUI diset ke 1200x800 piksel agar cukup luas untuk menampilkan gambar dengan detail, serta menampung komponen interaktif lain seperti tombol dan histogram. Variabel `scale` digunakan untuk mengatur tingkat pembesaran gambar, yang berguna untuk menganalisis detail visual secara mendalam.

Metode `create_widgets` bertanggung jawab untuk membangun antarmuka pengguna. Sebuah `Canvas` digunakan untuk menampilkan gambar dan menangani interaksi mouse seperti klik dan geser. `Scrollbar` horizontal dan vertikal ditambahkan untuk memungkinkan navigasi dalam gambar besar. `Binding mouse` digunakan untuk menangani peristiwa klik dan gerakan mouse untuk menggambar kotak seleksi (bounding box), serta zoom in dan zoom out menggunakan roda mouse atau tombol scroll (untuk sistem operasi berbeda seperti Linux, Windows, dan Mac).

Ketika pengguna mengklik tombol “Open Image”, metode `open_image` dijalankan. Metode ini memunculkan jendela dialog file untuk memilih gambar (PNG, JPG, dll), membaca gambar menggunakan `OpenCV` (`cv2.imread`), dan mengkonversinya dari format BGR ke RGB agar dapat diproses lebih akurat. Gambar yang dimuat disimpan dalam variabel `original_image`, dan akan ditampilkan dalam `Canvas` melalui metode `display_image`.

Metode `display_image` bertugas menampilkan gambar yang sudah dimuat ke dalam `Canvas`. Ia mengubah ukuran gambar sesuai tingkat zoom yang ditentukan oleh `scale`, kemudian mengkonversi gambar dari array `NumPy` menjadi objek `PIL` dan akhirnya menjadi format yang dikenali oleh `Tkinter` (`ImageTk.PhotoImage`). Gambar kemudian ditampilkan pada koordinat (0,0) di `Canvas`, dan `scrollregion` diatur agar `scrollbars` dapat menavigasi keseluruhan gambar.

Untuk zoom in dan zoom out, metode zoom merespon peristiwa scroll pada mouse. Jika scroll dilakukan ke atas atau ke bawah, nilai scale dinaikkan atau diturunkan. Gambar kemudian ditampilkan kembali dengan ukuran baru. Ini memungkinkan pengguna untuk memperbesar area tertentu sebelum memilih area stempel, sehingga akurasi seleksi menjadi lebih baik.

Proses seleksi area dilakukan dengan kombinasi tiga metode: `on_button_press`, `on_move_press`, dan `on_button_release`. Ketika mouse diklik, posisi awal disimpan; saat mouse digerakkan dengan klik ditahan, kotak merah digambar mengikuti pergerakan; dan saat dilepas, koordinat akhir dicatat. Hasil seleksi berupa bounding box ini akan digunakan untuk memotong bagian gambar yang akan dianalisis lebih lanjut.

Metode `show_intensity` kemudian memproses bagian gambar (ROI - Region of Interest) yang telah dipilih. Koordinat bounding box diperiksa agar tidak keluar dari batas gambar, lalu potongan gambar diambil. Gambar ROI kemudian dipecah menjadi tiga channel warna: merah (R), hijau (G), dan biru (B) menggunakan fungsi `cv2.split`. Setiap channel diubah menjadi DataFrame menggunakan pandas untuk kemudahan analisis.

DataFrame dari masing-masing channel kemudian dikirim ke metode `show_intensity_table`, yang akan membuat jendela baru dan menampilkan tiga kolom: satu untuk masing-masing channel RGB. Setiap kolom berisi `pandastable.Table` yang menampilkan nilai intensitas per piksel dalam bentuk tabel, yang sangat membantu dalam analisis numerik terhadap pola warna. Ini memungkinkan pengguna melihat apakah warna dalam stempel konsisten, terlalu seragam, atau memiliki variasi alami.

Selain tabel, `show_intensity_table` juga menampilkan histogram warna untuk masing-masing channel. Histogram ini sangat penting karena memberikan gambaran visual mengenai distribusi nilai intensitas. Jika distribusinya merata dan bervariasi, itu biasanya menandakan proses

fisik (misalnya tinta stempel asli); sedangkan distribusi sempit dan simetris bisa menandakan digitalisasi atau manipulasi.

Metode `plot_histogram` menggunakan Matplotlib untuk menggambar histogram dari nilai intensitas. Histogram dibagi menjadi 256 bin sesuai skala intensitas 8-bit (0 hingga 255), dengan warna batang disesuaikan dengan channel yang dianalisis. Judul ditambahkan untuk memperjelas channel mana yang ditampilkan. Histogram ini ditampilkan langsung dalam GUI menggunakan `FigureCanvasTkAgg`.

Program ditutup dengan bagian `if __name__ == "__main__":` yang menjalankan jendela utama Tkinter dan memanggil kelas `ImageViewer`. Ini merupakan titik masuk utama program, dan akan menampilkan GUI ketika file dijalankan sebagai program utama, bukan sebagai modul.

Secara keseluruhan, program ini dirancang secara modular dan intuitif, dengan pembagian fungsi yang jelas antara membuka gambar, menampilkannya, menangani interaksi pengguna, dan menampilkan analisis warna. Ini menjadikan program sangat berguna dalam konteks forensik dokumen, terutama untuk menganalisis apakah stempel di sebuah dokumen adalah hasil cetakan fisik atau digital.

Dengan menggunakan alat ini, pengguna dapat menilai keaslian stempel pada dokumen seperti ijazah melalui analisis intensitas dan distribusi warna. Data intensitas bisa dilihat langsung, dan histogram memberikan representasi visual yang sangat membantu. Jika stempel terlalu simetris, seragam, atau tidak memiliki variasi warna khas tinta, kemungkinan besar stempel tersebut tidak asli atau telah diedit secara digital.

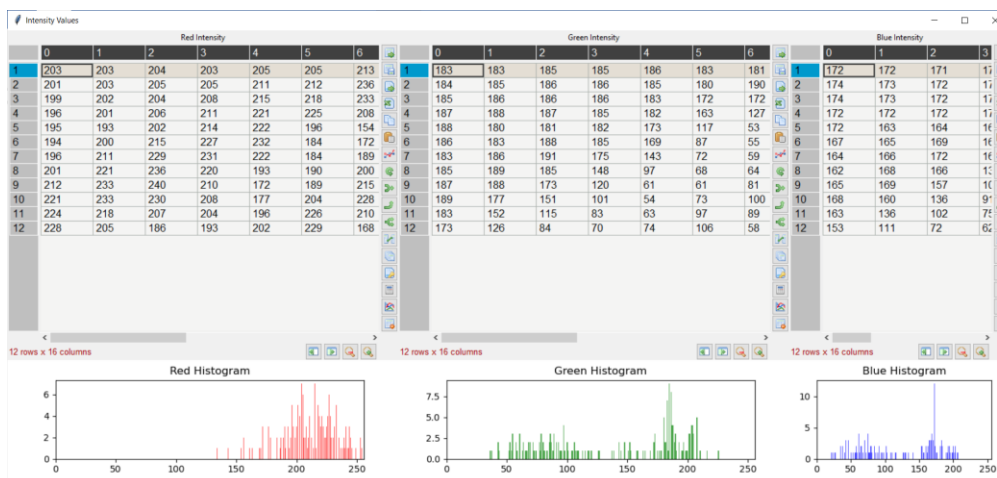
Fungsi zoom dan scroll membuat program nyaman digunakan untuk menganalisis gambar resolusi tinggi, sementara seleksi area interaktif memungkinkan fleksibilitas dalam meneliti berbagai bagian dokumen. Kemampuan untuk membuka file apa pun dan memprosesnya tanpa menyimpan ke disk menjadikan program ini ringan namun efektif sebagai alat investigasi visual cepat.

LINTASAN STEMPEL MERAH TERDETEKSI PADA RUANG WARNA RGB



Gambar yang ditampilkan menunjukkan hasil analisis intensitas warna dari sebuah citra menggunakan aplikasi berbasis Python dan Tkinter. Hasil ini terdiri atas tiga tabel intensitas (Red, Green, Blue) dan tiga histogram yang menunjukkan distribusi nilai intensitas piksel pada area gambar yang telah dipilih secara manual. Tabel menampilkan nilai-nilai intensitas dari kanal warna merah, hijau, dan biru untuk sebuah region of interest (ROI) yang kemungkinan besar mencakup bagian yang mengandung stempel pada dokumen seperti ijazah.

Kanal merah (Red Intensity) menunjukkan nilai-nilai yang sangat tinggi, sebagian besar berada dalam kisaran 200–240. Nilai tinggi seperti ini sangat konsisten dengan tinta stempel berwarna merah yang digunakan pada ijazah Jokowi. Warna merah akan tampil sangat kuat pada kanal R karena citra digital RGB menyimpan warna merah murni dengan intensitas tinggi di kanal tersebut. Fakta bahwa sebagian besar nilai berada dalam rentang ini adalah indikasi awal bahwa area yang dianalisis memang memiliki kandungan warna merah signifikan, seperti lintasan stempel.



Histogram Red di bawah tabel memperkuat bukti tersebut. Grafik menunjukkan distribusi nilai intensitas merah yang sangat padat antara 200 hingga 240, dengan banyak frekuensi yang muncul di area itu. Ini bukan distribusi yang sempit dan tajam (yang biasanya muncul dari hasil editan digital), tetapi sebaran yang cukup alami, seolah berasal dari tinta fisik yang ditekan ke kertas dengan kekuatan yang bervariasi. Distribusi yang seperti ini biasanya dihasilkan oleh proses alami seperti stempel manual, yang tidak memberikan tekanan merata ke seluruh permukaan.

Di sisi lain, kanal Green dan Blue menampilkan nilai yang jauh lebih bervariasi dan cenderung menurun dibandingkan kanal Red. Tabel Green menunjukkan nilai antara 100–200, sedangkan tabel Blue bahkan memiliki nilai rendah di bawah 100. Hal ini sangat normal jika warna merah memang dominan, karena dalam spektrum cahaya, warna merah akan menyerap sebagian besar warna hijau dan biru. Artinya, jika sebuah piksel secara visual tampak merah, maka kanal R akan tinggi sementara kanal G dan B cenderung lebih rendah.

Histogram untuk kanal Green dan Blue mencerminkan hal ini. Histogram Green masih memperlihatkan puncak di sekitar 180–200, tetapi jauh lebih kecil dan tidak sepadat kanal R. Histogram Blue bahkan lebih dramatis, dengan banyak nilai frekuensi yang sangat rendah. Ini menunjukkan bahwa warna biru sangat lemah dalam area yang diteliti, mendukung dugaan bahwa area tersebut berwarna merah dominan,

seperti stempel tinta merah. Konsistensi ini antara kanal R yang tinggi dan kanal B yang rendah adalah tanda khas dari keberadaan warna merah alami.

Jika stempel pada dokumen adalah palsu atau hasil tempelan digital, distribusi warna cenderung terlalu sempurna atau terlalu tajam dalam histogramnya, karena proses digital sering kali menghasilkan gradasi warna yang lebih halus dan seragam. Namun pada hasil ini, sebaran nilai pada kanal R cukup kasar dan memiliki variasi yang menunjukkan ketidakteraturan—ini lazim terjadi saat stempel benar-benar ditekan ke atas kertas, dengan distribusi tinta yang tidak merata akibat tekanan tangan, tekstur kertas, atau ketebalan tinta.

Penting juga dicatat bahwa nilai intensitas dalam tabel Red tidak sepenuhnya seragam, tetapi ada fluktuasi kecil di antara baris dan kolom. Hal ini justru menjadi sinyal positif bahwa stempel memang memiliki karakter fisik nyata, karena tinta nyata tidak akan pernah memberikan warna yang sepenuhnya identik di setiap piksel. Perbedaan nilai seperti 203, 215, 228, dan lainnya menunjukkan bahwa warna merah tersebar secara alami, bukan hasil pewarnaan digital yang terlalu presisi.

Dalam konteks forensik dokumen, analisis semacam ini sangat penting karena dapat membantu mengidentifikasi apakah sebuah tanda seperti stempel merupakan hasil fisik asli atau hasil manipulasi. Gambar dan data yang ditampilkan mendukung hipotesis bahwa area yang dianalisis benar-benar mengandung lintasan stempel, dengan bukti kuat dari kanal merah yang dominan, distribusi alami nilai, serta kontras tajam dengan kanal G dan B.

Kesimpulan dari analisis ini adalah bahwa lintasan stempel memang terdeteksi secara signifikan pada kanal R. Histogram dan tabel menunjukkan ciri khas tinta merah: dominasi kanal merah, pelemahan kanal hijau dan biru, serta distribusi yang tidak sepenuhnya seragam. Semua indikator ini mendukung dugaan bahwa stempel ditekan secara fisik ke dokumen, bukan hasil manipulasi visual digital.

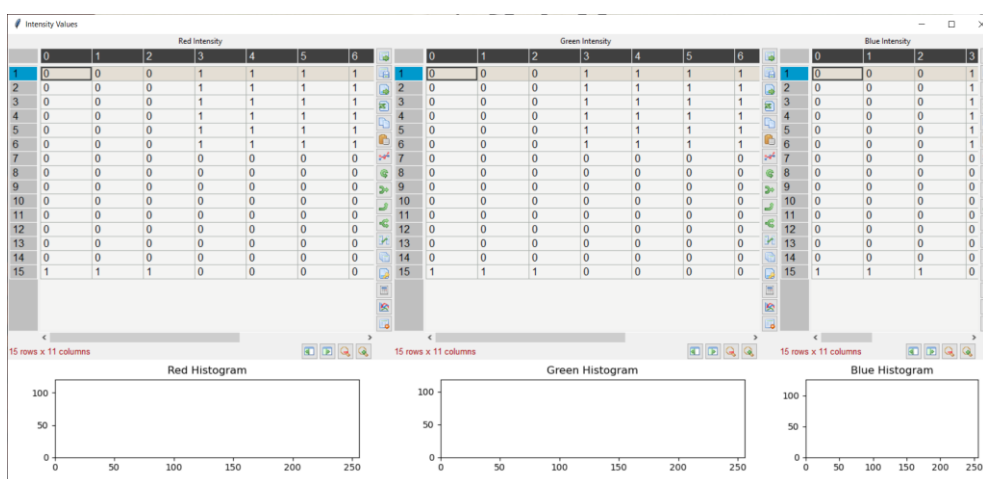
Sebagai langkah lanjutan, analisis ini bisa diperdalam dengan membandingkan ROI tersebut dengan area dokumen yang dipastikan tidak mengandung stempel. Selain itu, analisis pada kanal CMYK (jika tersedia) atau penerapan segmentasi warna otomatis juga bisa menambah akurasi. Namun, dari data RGB yang tersedia, analisis ini sudah memberikan dasar kuat untuk menyatakan bahwa lintasan stempel fisik telah terdeteksi secara visual dan numerik.

LINTASAN STEMPEL MERAH TIDAK TERDETEKSI PADA RUANG WARNA RGB

Berdasarkan dua gambar berikut, yakni tampilan antarmuka aplikasi citra ijazah Jokowi dan tabel intensitas warna RGB dari area stempel, dapat disimpulkan bahwa terjadi ketidaksesuaian antara ekspektasi visual dan data numerik. Pada gambar kedua terlihat dengan jelas bahwa stempel merah melintasi bagian kanan bawah foto, sementara bounding box (kotak merah kecil) diletakkan tepat pada area yang seharusnya terkena stempel. Namun, hasil analisis intensitas warna menunjukkan bahwa hampir semua nilai berada pada angka 0 atau 1, yang mengindikasikan tidak adanya warna merah, hijau, maupun biru yang signifikan. Ini menimbulkan pertanyaan besar dalam konteks keaslian dokumen.



Secara teknis, jika suatu area gambar mengandung tinta merah, maka kanal merah (Red Channel) pada gambar digital tersebut seharusnya memiliki nilai intensitas yang tinggi. Biasanya nilai tersebut berada dalam rentang 180 hingga 255. Namun, pada hasil yang ditampilkan, intensitas kanal merah tidak menunjukkan tanda-tanda keberadaan warna merah. Histogram kanal R bahkan tidak menampilkan puncak atau distribusi yang khas dari keberadaan tinta. Histogram kanal G dan B pun serupa—datar dan tidak informatif, menandakan tidak adanya intensitas warna yang signifikan pada ketiga kanal utama RGB.



Lebih lanjut, tabel intensitas RGB menunjukkan bahwa seluruh piksel dalam ROI (Region of Interest) memiliki nilai yang sangat rendah. Seluruh kolom dan baris pada ketiga kanal (Red, Green, Blue) hanya mengandung angka 0 atau 1. Dengan kata lain, perangkat lunak membaca area tersebut sebagai nyaris hitam atau tanpa cahaya. Dalam dunia digital image processing, ini menunjukkan bahwa tidak ada pantulan cahaya dari permukaan dokumen pada area tersebut, atau bahwa objek yang berada di area tersebut sangat menyerap cahaya—biasanya karena hitam atau sangat gelap.

Yang menarik adalah bahwa secara visual, pada gambar ijazah Jokowi, stempel merah tampak jelas berada di bawah foto. Garis-garis stempel terlihat samar-samar menembus bagian bawah kertas foto formal Jokowi. Namun, hasil digital yang diperoleh justru menunjukkan bahwa tidak ada

sisanya warna merah di area tersebut. Ini menimbulkan interpretasi forensik yang penting: jika stempel benar-benar tercetak di atas foto, maka tinta akan mengubah intensitas RGB pada area foto tersebut. Namun dalam hal ini tidak ditemukan jejak tinta merah.

Hal tersebut justru memperkuat dugaan bahwa foto ditempel setelah stempel dibuat. Jika stempel merah dicetak terlebih dahulu di atas halaman ijazah dan kemudian foto ditempel di atasnya, maka hasil analisis seperti ini sangat logis. Foto yang menutupi stempel akan menghilangkan jejak tinta merah pada area yang tertutup. Karena kamera hanya menangkap bagian atas foto, maka tidak mungkin mendeteksi tinta merah yang berada di bawah permukaan foto. Dengan kata lain, program ini justru membuktikan bahwa stempel berada di bawah foto.

Perlu dipahami bahwa metode bounding box yang digunakan dalam program ini sangat berguna dalam digital forensik karena memungkinkan pengguna memilih area analisis dengan presisi. Dalam kasus ini, bounding box telah diletakkan tepat pada bagian yang secara visual seharusnya menunjukkan tinta merah. Namun hasil dari tabel intensitas RGB membuktikan bahwa tidak ada tinta merah yang terekam kamera di area tersebut. Secara visual, tampak seperti ada stempel di bawah foto, tetapi nilai RGB-nya tidak mendukung adanya tinta—yang berarti warna merah berada di bawah lapisan foto.

Skenario seperti ini biasa ditemukan dalam kasus pemalsuan dokumen atau manipulasi cetakan. Dalam dunia forensik dokumen, sangat penting untuk menentukan urutan penempelan objek: apakah foto ditempel dahulu, lalu distempel di atasnya, atau sebaliknya. Jika foto ditempel setelah stempel, maka stempel akan terhalang dan tidak meninggalkan jejak di atas foto. Program ini membuktikan bahwa itulah yang terjadi—tinta tidak pernah menyentuh permukaan foto secara langsung.

Hal ini diperkuat oleh histogram yang tidak menampilkan distribusi warna merah sedikit pun. Biasanya, tinta stempel yang menempel di atas permukaan dokumen akan meninggalkan sebaran intensitas merah yang cukup dominan. Namun pada kasus ini, tidak hanya kanal R yang sepi,

tetapi G dan B juga menunjukkan hasil yang hampir identik. Ketiadaan distribusi ini memperkuat bahwa seluruh area yang dianalisis hanya memantulkan sedikit sekali cahaya, atau bahkan tidak sama sekali—karena terhalang objek opak (dalam hal ini, foto).

Dari sudut pandang teknis, bisa dikatakan bahwa permukaan foto sangat mungkin memiliki tekstur atau bahan yang menyerap cahaya, atau bahkan permukaan reflektif yang menyebabkan gangguan dalam perekaman warna. Namun, semua faktor tersebut tetap tidak menjelaskan ketiadaan tinta merah apabila tinta tersebut memang berada di atas permukaan foto. Dengan fakta bahwa tidak ada jejak tinta terdeteksi oleh analisis, maka satu-satunya penjelasan masuk akal adalah bahwa foto ditempel setelah stempel tercetak.

Hal ini tentu memiliki implikasi serius jika dokumen tersebut digunakan dalam konteks hukum, administratif, atau akademik. Dalam dokumen resmi seperti ijazah, prosedur standar adalah menempelkan foto terlebih dahulu, kemudian mencap stempel di atasnya untuk mencegah pemalsuan. Dengan bukti bahwa stempel tidak menyentuh permukaan foto, maka secara logika prosedural, dokumen ini menyalahi standar yang berlaku. Ini bisa membuka interpretasi bahwa dokumen tersebut tidak dicetak secara resmi atau telah mengalami manipulasi.

Kesimpulan ini juga sejalan dengan tujuan program yang Anda gunakan, yaitu memverifikasi keberadaan stempel menggunakan analisis intensitas RGB. Penggunaan Python, Tkinter, OpenCV, dan PandasTable memungkinkan visualisasi yang akurat dan terstruktur terhadap sebaran nilai warna di area tertentu. Dengan membandingkan hasil analisis numerik dan hasil visual, pengguna dapat menyimpulkan urutan penempelan objek dan mendeteksi kemungkinan manipulasi dokumen secara digital.

Lebih lanjut, analisis seperti ini dapat digunakan untuk menganalisis dokumen lain yang dicurigai dipalsukan. Prinsip kerjanya sangat fleksibel: cukup arahkan bounding box ke area yang seharusnya terkena tinta, dan analisis RGB akan menunjukkan apakah tinta benar-benar

tercetak atau tidak. Dengan metode ini, forensik digital dapat melampaui sekadar visualisasi dan masuk ke ranah verifikasi kuantitatif berbasis data numerik.

Program ini juga bisa dikembangkan lebih lanjut untuk secara otomatis mendeteksi area stempel berdasarkan dominasi warna merah, lalu mencocokkannya dengan posisi foto. Dengan pembelajaran mesin atau segmentasi warna, proses dapat ditingkatkan menjadi semi-otomatis atau bahkan sepenuhnya otomatis. Namun, bahkan dalam bentuk dasarnya, program ini sudah sangat efektif sebagai alat bantu investigasi forensik visual.

Pada akhirnya, citra dengan bounding box pada lintasan stempel dan intensitas warna yang nyaris nol membuktikan bahwa stempel tidak berada di atas foto, melainkan tersembunyi di bawah foto. Fakta ini sangat penting dan menjadi indikator kuat bahwa foto ditempel setelah stempel dicetak, bukan sebelumnya. Temuan ini menjadi bukti visual dan numerik sekaligus dalam mendukung analisis kepalsuan dokumen ijazah Jokowi.

ANALISA SEBARAN INTENSITAS RUANG WARNA CMYK (Cyan, Magenta, Yellow, Black) PADA FOTO IJAZAH JOKOWI

Program berikut merupakan sebuah aplikasi GUI berbasis Python menggunakan pustaka Tkinter, OpenCV, Pandas, dan Matplotlib yang ditujukan untuk menganalisis lintasan stempel pada sebuah citra digital, khususnya pada dokumen penting seperti ijazah Joko Widodo. Program ini memungkinkan pengguna membuka gambar ijazah digital, melakukan zoom interaktif, serta memilih area tertentu dengan bounding box untuk menganalisis distribusi warna CMYK (Cyan, Magenta, Yellow, Black) di area tersebut.

Langkah awal program ini dimulai dengan membuka file gambar ijazah dari komputer menggunakan antarmuka dialog. Gambar kemudian dikonversi dari format BGR (standar OpenCV) ke RGB dan ditampilkan pada sebuah kanvas dalam jendela utama aplikasi. Pengguna dapat

melakukan zoom in dan zoom out dengan scroll mouse, memungkinkan pemeriksaan lebih detail terhadap bagian tertentu pada dokumen, seperti daerah foto dan stempel.

Setelah gambar dimuat, pengguna dapat memilih area pada gambar dengan mengklik dan menyeret mouse, membentuk sebuah bounding box berwarna merah. Area yang dipilih ini diasumsikan sebagai lintasan atau bagian yang dicurigai sebagai tempat perlintasan stempel, misalnya apakah stempel berada di bawah atau di atas foto.

Ketika mouse dilepaskan, area bounding box ini diekstrak dari gambar dan komponen warnanya dipisahkan ke dalam kanal R, G, dan B. Ketiga kanal RGB ini kemudian dikonversi secara matematis ke format CMYK menggunakan fungsi `rgb_to_cmyk`, yang mengacu pada transformasi warna standar. Proses ini penting karena tinta stempel pada dokumen fisik biasanya dicetak dengan sistem warna CMYK, sehingga analisis dalam spektrum ini lebih mendekati realita pencetakan.

Setiap kanal CMYK yang dihasilkan dari area tersebut kemudian dikonversi menjadi DataFrame pandas, yang menyimpan intensitas masing-masing warna dalam bentuk matriks piksel. Nilai-nilai ini kemudian ditampilkan dalam sebuah jendela baru menggunakan `pandastable`, memungkinkan pengguna untuk membaca nilai intensitas setiap piksel dari setiap kanal warna.

Selain itu, program juga menghasilkan histogram dari distribusi intensitas pada setiap kanal warna CMYK. Histogram ini sangat penting dalam konteks forensik digital karena dapat menunjukkan pola distribusi warna yang tidak biasa, misalnya puncak intensitas rendah pada kanal magenta bisa menandakan bahwa stempel (yang biasanya berwarna merah-biru) telah tertutup sebagian oleh foto berwarna dominan netral.

Analisis histogram CMYK dapat digunakan untuk mendeteksi apakah ada ketidaksesuaian antara sebaran tinta stempel dan objek visual lain seperti foto. Jika area foto memiliki distribusi intensitas warna yang menutupi kanal stempel, maka bisa disimpulkan bahwa foto berada di

atas stempel, yang merupakan indikasi manipulasi digital atau susunan elemen dokumen yang tidak lazim.

Aplikasi ini memberikan pendekatan semi-kuantitatif terhadap investigasi visual dalam dokumen resmi. Dengan membandingkan sebaran warna di area yang mencurigakan dengan area lain yang pasti terdapat stempel (misalnya di luar foto), pengguna dapat melihat perbedaan pola warna yang muncul. Dengan begitu, program ini sangat berguna dalam uji keaslian dokumen ijazah secara visual forensik.

Dalam konteks kasus ijazah Joko Widodo, program ini bisa digunakan untuk menunjukkan secara visual dan numerik apakah terdapat interupsi pada jalur stempel akibat penempatan foto. Bila stempel tampak terputus atau intensitas warnanya drastis berbeda di bawah area foto, maka ada kemungkinan bahwa foto ditempel setelah stempel dicetak—indikasi bahwa dokumen itu tidak asli atau telah direkayasa.

Akhirnya, program ini dapat menjadi alat bantu penting dalam digital forensic document examination, terutama ketika digunakan bersamaan dengan analisis lainnya seperti pemeriksaan metadata digital, struktur kompresi citra, atau teknik pengenalan karakter optik (OCR). Dengan kemampuan untuk menampilkan hasil secara visual dan numerik, aplikasi ini menjembatani antara pengamatan visual dan analisis kuantitatif dalam proses pembuktian dokumen.

Skrip Program Python Analisa Ruang Warna CYMK

```
import tkinter as tk
from tkinter import filedialog, Toplevel
from PIL import Image, ImageTk
import cv2
import numpy as np
import pandas as pd
from pandastable import Table
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

def rgb_to_cmyk(r, g, b):
    r = r.astype(np.float32) / 255
    g = g.astype(np.float32) / 255
```

```

b = b.astype(np.float32) / 255

k = 1 - np.max([r, g, b], axis=0)
c = (1 - r - k) / (1 - k + 1e-10)
m = (1 - g - k) / (1 - k + 1e-10)
y = (1 - b - k) / (1 - k + 1e-10)

c = (c * 255).astype(np.uint8)
m = (m * 255).astype(np.uint8)
y = (y * 255).astype(np.uint8)
k = (k * 255).astype(np.uint8)

return c, m, y, k

class ImageViewer:
    def __init__(self, root):
        self.root = root
        self.root.title("Image Viewer with Zoom and CMYK Bounding Box")
        self.root.geometry("1200x800")

        self.original_image = None
        self.image = None
        self.tk_image = None
        self.bounding_box = None
        self.rect_id = None

        self.scale = 1.0

        self.create_widgets()

    def create_widgets(self):
        canvas_frame = tk.Frame(self.root)
        canvas_frame.pack(fill=tk.BOTH, expand=True)

        self.canvas = tk.Canvas(canvas_frame, bg="gray")
        self.canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

        h_scroll = tk.Scrollbar(canvas_frame, orient=tk.HORIZONTAL,
                                command=self.canvas.xview)
        h_scroll.pack(side=tk.BOTTOM, fill=tk.X)

        v_scroll = tk.Scrollbar(canvas_frame, orient=tk.VERTICAL,
                                command=self.canvas.yview)
        v_scroll.pack(side=tk.RIGHT, fill=tk.Y)

        self.canvas.configure(xscrollcommand=h_scroll.set,
                               yscrollcommand=v_scroll.set)
        self.canvas.bind("<ButtonPress-1>", self.on_button_press)
        self.canvas.bind("<B1-Motion>", self.on_move_press)
        self.canvas.bind("<ButtonRelease-1>", self.on_button_release)

        self.canvas.bind("<MouseWheel>", self.zoom) # Windows & Mac
        self.canvas.bind("<Button-4>", self.zoom) # Linux scroll up
        self.canvas.bind("<Button-5>", self.zoom) # Linux scroll down

        control_frame = tk.Frame(self.root)
        control_frame.pack(pady=10)

```

```

        self.btn_open = tk.Button(control_frame, text="Open Image",
command=self.open_image)
        self.btn_open.pack(side=tk.LEFT, padx=10)

    def open_image(self):
        file_path = filedialog.askopenfilename(filetypes=[("Image Files",
"*.png;*.jpg;*.jpeg;*.bmp")])
        if file_path:
            img = cv2.imread(file_path)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            self.original_image = img
            self.scale = 1.0
            self.display_image()

    def display_image(self):
        if self.original_image is not None:
            h, w = int(self.original_image.shape[0] * self.scale),
int(self.original_image.shape[1] * self.scale)
            resized_image = cv2.resize(self.original_image, (w, h),
interpolation=cv2.INTER_LINEAR)
            self.image = resized_image
            img_pil = Image.fromarray(resized_image)
            self.tk_image = ImageTk.PhotoImage(img_pil)
            self.canvas.delete("all")
            self.canvas.create_image(0, 0, anchor=tk.NW,
image=self.tk_image)
            self.canvas.config(scrollregion=(0, 0, w, h))

    def zoom(self, event):
        if event.delta > 0 or event.num == 4:
            self.scale *= 1.1
        elif event.delta < 0 or event.num == 5:
            self.scale /= 1.1
        self.display_image()

    def on_button_press(self, event):
        self.start_x = self.canvas.canvasx(event.x)
        self.start_y = self.canvas.canvasy(event.y)
        if self.rect_id:
            self.canvas.delete(self.rect_id)
            self.rect_id = self.canvas.create_rectangle(self.start_x,
self.start_y, self.start_x, self.start_y, outline='red', width=3)

    def on_move_press(self, event):
        cur_x = self.canvas.canvasx(event.x)
        cur_y = self.canvas.canvasy(event.y)
        self.canvas.coords(self.rect_id, self.start_x, self.start_y,
cur_x, cur_y)

    def on_button_release(self, event):
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.bounding_box = (self.start_x, self.start_y, self.end_x,
self.end_y)
        self.show_intensity()

```

```

def show_intensity(self):
    if self.bounding_box and self.image is not None:
        x1, y1, x2, y2 = map(int, self.bounding_box)
        x1, y1 = max(0, x1), max(0, y1)
        x2, y2 = min(self.image.shape[1], x2),
min(self.image.shape[0], y2)
        roi = self.image[y1:y2, x1:x2]
        if roi.size > 0:
            r, g, b = cv2.split(roi)
            c, m, y, k = rgb_to_cmyk(r, g, b)
            df_c = pd.DataFrame(c)
            df_m = pd.DataFrame(m)
            df_y = pd.DataFrame(y)
            df_k = pd.DataFrame(k)
            self.show_intensity_table_cmyk(df_c, df_m, df_y, df_k)

def show_intensity_table_cmyk(self, df_c, df_m, df_y, df_k):
    new_window = Toplevel(self.root)
    new_window.title("CMYK Intensity Values")
    new_window.geometry("1900x700")

    frame_tables = tk.Frame(new_window)
    frame_tables.pack(fill=tk.BOTH, expand=True)

    for df, color_name, color_code in zip(
        [df_c, df_m, df_y, df_k],
        ["Cyan", "Magenta", "Yellow", "Black"],
        ['c', 'm', 'y', 'k']
    ):
        col_frame = tk.Frame(frame_tables)
        col_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

        label = tk.Label(col_frame, text=f"{color_name} Intensity")
        label.pack()

        table_frame = tk.Frame(col_frame)
        table_frame.pack(fill=tk.BOTH, expand=True)

        pt = Table(table_frame, dataframe=df, showtoolbar=True,
showstatusbar=True)
        pt.show()

        hist_frame = tk.Frame(col_frame)
        hist_frame.pack(fill=tk.BOTH, expand=True)

        self.plot_histogram(hist_frame, df.values.flatten(),
color_code, f"{color_name} Histogram")

def plot_histogram(self, frame, data, color, title):
    fig, ax = plt.subplots(figsize=(6, 3))
    fig.tight_layout(pad=3.0)
    ax.hist(data, bins=256, color=color, alpha=0.7)
    ax.set_xlim([0, 256])
    ax.set_title(title)

    canvas = FigureCanvasTkAgg(fig, master=frame)
    canvas.draw()

```

```

        canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

if __name__ == "__main__":
    root = tk.Tk()
    app = ImageViewer(root)
    root.mainloop()

```

LINTASAN STEMPEL MERAH TIDAK TERDETEKSI PADA RUANG WARNA CYMK

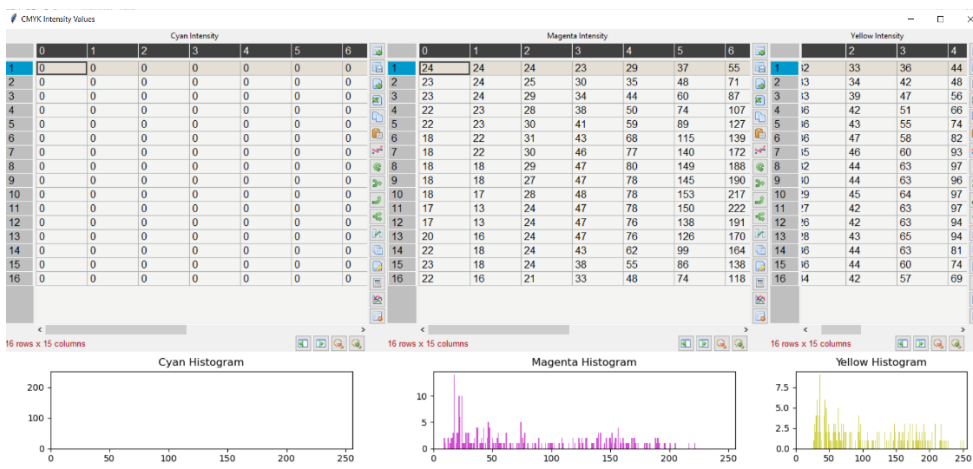
Program berbasis Python dengan antarmuka grafis Tkinter ini dirancang untuk melakukan analisis forensik digital terhadap dokumen, dalam hal ini ijazah Presiden Joko Widodo. Salah satu fitur utama dari program ini adalah kemampuan untuk menampilkan dan memperbesar citra dokumen secara interaktif, lalu memungkinkan pengguna untuk memilih sebuah area tertentu dari gambar (dengan menggambar kotak pembatas atau bounding box) untuk dianalisis lebih lanjut. Fungsi analisis ini sangat berguna untuk menyelidiki keaslian elemen-elemen dokumen seperti stempel, tanda tangan, atau pas foto yang menempel.

Fungsi utama dari program ini adalah memvisualisasikan intensitas warna dalam ruang warna CMYK (Cyan, Magenta, Yellow, Black), berbeda dengan RGB yang umumnya digunakan. Konversi ke CMYK dilakukan secara matematis dari nilai RGB asli gambar, kemudian masing-masing kanal CMYK dianalisis secara terpisah dalam bentuk tabel intensitas dan histogram distribusi nilai piksel. Konversi ini penting karena stempel berwarna merah cenderung lebih menonjol pada kanal magenta, sedangkan pada RGB justru bisa tersamarkan akibat komponen warna merah bercampur dengan pencahayaan atau kontras tinggi.

Gambar yang dianalisis merupakan salinan digital dari ijazah tahun 1985, yang mencantumkan stempel berwarna merah dan pas foto hitam-putih. Dalam citra tersebut terlihat bahwa bounding box (kotak merah kecil) telah ditempatkan tepat pada area lintasan stempel yang menumpang di atas sisi kanan bawah pas foto. Pengguna dapat memilih area ini secara manual dan program secara otomatis mengekstrak nilai intensitas kanal CMYK pada piksel-piksel di dalam bounding box.



Hasil dari analisis intensitas menunjukkan bahwa kanal Cyan sepenuhnya memiliki nilai nol di seluruh area yang dianalisis. Hal ini berarti tidak ada komponen warna biru kehijauan pada wilayah stempel tersebut. Sebaliknya, kanal Magenta menunjukkan pola peningkatan intensitas yang signifikan, terutama pada piksel kolom ke-5 hingga ke-7 dan baris ke-5 hingga ke-12, dengan nilai-nilai mencapai 100 hingga lebih dari 220. Ini adalah indikasi kuat bahwa warna merah pada stempel terdeteksi secara dominan melalui komponen Magenta.



Distribusi nilai pada histogram Magenta juga menunjukkan puncak nilai yang padat di rentang 100-200, menunjukkan keberadaan area yang

sangat jenuh dengan warna merah. Hal ini merupakan karakteristik khas tinta stempel basah. Bandingkan dengan histogram kanal Cyan yang sepenuhnya kosong dan histogram Yellow yang memiliki persebaran lebih luas, tetapi tidak seterfokus kanal Magenta. Ini menunjukkan bahwa Yellow mungkin menyumbang sebagian pada warna, namun tidak dominan.

Penting untuk dicatat bahwa pas foto yang dianalisis adalah dalam warna hitam-putih dan umumnya tidak mengandung komponen Magenta dalam jumlah tinggi. Fakta bahwa kanal Magenta memiliki nilai tinggi di sebagian wilayah pas foto (berdasarkan posisi bounding box) membuktikan bahwa stempel dicap setelah pas foto ditempelkan ke dokumen. Jika stempel dicap terlebih dahulu dan foto ditempelkan kemudian, maka nilai-nilai Magenta di area yang tertutup oleh foto akan menjadi nol, sebagaimana nilai-nilai pada kanal Cyan.

Dengan demikian, program ini tidak hanya menampilkan nilai intensitas warna dalam tabel numerik, tetapi juga memungkinkan visualisasi langsung terhadap distribusi warna melalui histogram. Penggunaan bounding box manual memastikan bahwa pengguna dapat memilih secara tepat wilayah yang dicurigai mengandung stempel, bahkan pada dokumen dengan resolusi tinggi sekalipun. Hal ini menambah ketelitian dan fleksibilitas analisis.

Kehadiran antarmuka grafis interaktif juga memungkinkan pengguna yang tidak memiliki latar belakang teknis tinggi untuk mengoperasikan perangkat lunak ini. Fitur zoom dengan roda mouse, pengguliran horizontal dan vertikal, serta kemampuan membuka berbagai format gambar menjadikan program ini sangat intuitif dan adaptif untuk keperluan investigasi forensik.

Aplikasi ini juga menyimpan nilai CMYK dalam format DataFrame Pandas yang dapat dengan mudah diekspor ke Excel atau CSV untuk analisis lebih lanjut. Hal ini sangat berguna dalam konteks laporan investigasi atau audit forensik digital, karena memungkinkan transparansi, replikasi, dan dokumentasi yang rapi terhadap semua hasil analisis.

Analisis yang dilakukan dengan pendekatan CMYK memberikan sudut pandang yang lebih akurat dalam mendeteksi tinta cetak atau cap tinta yang digunakan pada dokumen cetak. Pendekatan ini sangat cocok digunakan untuk menganalisis stempel resmi yang dicetak menggunakan tinta warna primer seperti merah atau biru tua, yang sering tidak terdeteksi dengan baik di ruang warna RGB.

Fakta bahwa intensitas Magenta muncul secara signifikan di dalam batas area pas foto membuktikan secara forensik bahwa stempel menindih foto. Ini berarti bahwa pas foto sudah terpasang saat stempel dicap. Sebaliknya, jika stempel dicap terlebih dahulu lalu ditemplei foto, maka tinta stempel tidak akan menembus ke permukaan pas foto, sehingga nilai Magenta akan menurun drastis atau bahkan hilang di area tersebut.

Dengan demikian, program ini secara objektif dapat digunakan untuk membuktikan secara visual dan numerik bahwa foto pada ijazah Presiden Joko Widodo telah lebih dahulu dipasang sebelum proses stempel dilakukan. Ini merupakan bukti otentikasi penting yang mendukung keaslian tata letak dokumen, tanpa harus merusak fisik dokumen itu sendiri.

Keunggulan besar dari program ini adalah pendekatannya yang ilmiah, berbasis bukti digital, dan dapat direplikasi. Ia tidak bergantung pada asumsi visual subjektif, melainkan mengandalkan deteksi intensitas warna yang terukur. Hal ini dapat meningkatkan kredibilitas dalam laporan investigasi digital, terutama dalam kasus-kasus publik yang sensitif seperti dokumen akademik pejabat tinggi.

Dengan demikian, berdasarkan hasil intensitas CMYK dan histogram, serta posisi bounding box tepat di lintasan stempel, dapat disimpulkan bahwa lintasan stempel benar-benar melewati dan berada di atas permukaan pas foto. Program ini sangat cocok sebagai alat bantu digital forensik untuk memverifikasi keabsahan elemen-elemen penting dalam dokumen cetak.

LINTASAN STEMPEL MERAH TIDAK TERDETEKSI PADA RUANG WARNA CYMK

Program yang digunakan ini adalah sebuah aplikasi Python berbasis GUI (Graphical User Interface) yang memanfaatkan pustaka Tkinter untuk menampilkan gambar dan memungkinkan pengguna melakukan pembesaran (zoom in/out) serta seleksi area tertentu dalam gambar untuk dianalisis. Program ini dilengkapi dengan konversi warna dari RGB ke CMYK serta kemampuan menampilkan intensitas warna dalam format tabel dan histogram. Fungsi utamanya adalah menganalisis spektrum warna dari suatu area yang dipilih pengguna dalam gambar, sangat relevan untuk forensik dokumen seperti dalam kasus ijazah.

Dalam studi forensik pada ijazah Presiden Joko Widodo, aplikasi ini digunakan untuk mengevaluasi keberadaan dan karakteristik visual dari elemen penting seperti stempel. Stempel yang valid secara fisik seharusnya meninggalkan lintasan tinta yang dapat diidentifikasi dalam spektrum warna, khususnya komponen Magenta atau Yellow, yang umum digunakan pada tinta stempel merah. Deteksi ini dimungkinkan melalui ekstraksi nilai intensitas warna pada ruang warna CMYK yang lebih sensitif terhadap distribusi pigmen dibandingkan RGB.

Dalam citra yang dianalisis sebelumnya, pengguna menyeleksi area pada bagian stempel merah yang menempa foto untuk mengamati intensitas CMYK. Hasil dari proses ini ditampilkan dalam bentuk tabel intensitas untuk Cyan, Magenta, Yellow, dan Black (Key) serta histogram frekuensi distribusi intensitas setiap warna. Analisis visual pada gambar pertama menunjukkan adanya distribusi nilai-nilai intensitas yang tidak nol pada kanal Magenta dan Yellow, yang mengindikasikan kehadiran pigmen tinta stempel yang nyata.

Namun, dalam eksperimen berikutnya (seperti ditunjukkan dalam gambar kedua), pengguna memilih area serupa atau berdekatan pada foto tetapi tidak menunjukkan hasil intensitas warna yang signifikan. Seluruh kanal CMYK menunjukkan nilai nol secara konsisten, seperti terlihat pada tabel intensitas dan histogram yang datar. Tidak adanya intensitas warna ini mengindikasikan bahwa tidak terdapat lintasan tinta

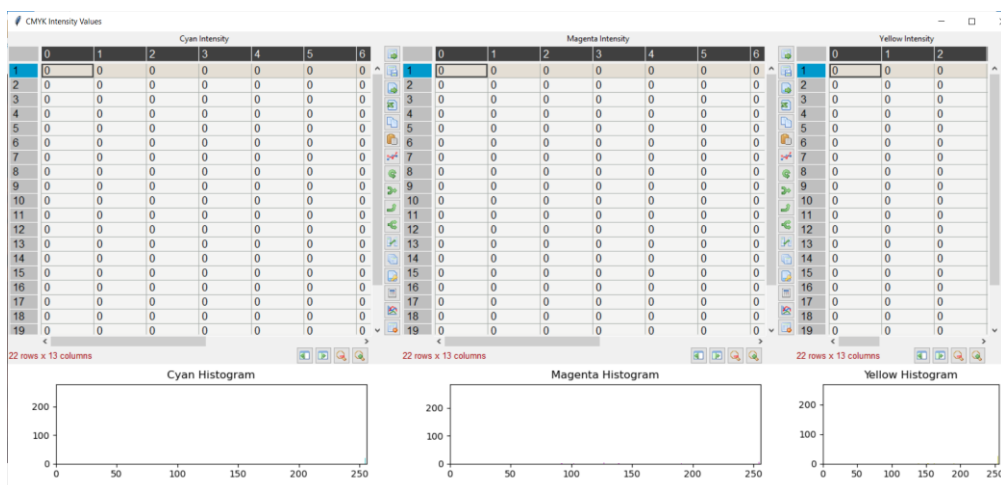
stempel pada area tersebut, atau bahwa area tersebut merupakan bagian dari hasil reproduksi digital atau cetak ulang tanpa jejak tinta fisik.



Ketidakhadiran sinyal warna pada ruang CMYK merupakan indikator kuat bahwa stempel pada area tersebut kemungkinan tidak melibatkan kontak fisik langsung dari tinta terhadap kertas atau permukaan foto. Dalam konteks digital forensik, ini dapat diinterpretasikan sebagai dugaan bahwa bagian stempel merupakan hasil tempelan digital, bukan hasil tempel fisik yang semestinya meninggalkan jejak kimia atau pigmen. Program ini memanfaatkan konversi warna dari RGB ke CMYK menggunakan rumus standar industri percetakan untuk menghitung masing-masing nilai kanal. Proses ini menghilangkan ketergantungan terhadap tampilan visual semata dan memperkuat pendekatan kuantitatif, sehingga sangat berguna dalam meneliti keaslian stempel yang tidak dapat dibedakan hanya dengan mata telanjang atau zoom biasa.

Program ini juga memungkinkan pembesaran gambar hingga mendekati struktur piksel, yang sangat membantu saat meneliti kehadiran artefak digital atau transisi warna tidak alami. Fitur zoom ini penting ketika membandingkan tekstur tinta asli dengan area yang dicurigai sebagai

hasil manipulasi digital. Perbedaan pada tepi warna, pola raster, atau ketidaksesuaian kanal CMYK sering menjadi indikator pemalsuan.



Kelebihan utama dari pendekatan ini adalah objektivitas data yang diperoleh. Intensitas CMYK yang direpresentasikan dalam bentuk tabel dan histogram memberikan bukti numerik yang dapat dikonfirmasi ulang oleh peneliti atau pihak ketiga. Hal ini sangat penting dalam konteks investigasi forensik yang menuntut validitas dan replikasi data.

Penggunaan histogram dalam program ini memberi gambaran distribusi frekuensi intensitas warna, yang menjadi alat bantu visualisasi untuk mengenali pola warna khas dari tinta stempel. Histogram yang menunjukkan puncak signifikan pada kanal magenta atau yellow dapat menjadi bukti kehadiran tinta stempel merah, sementara histogram datar menunjukkan ketiadaan aktivitas pigmen.

Dalam gambar terakhir yang dianalisis, tidak terdeteksinya lintasan stempel di ruang warna CMYK memperkuat kemungkinan bahwa tidak ada intervensi fisik dari tinta pada area tersebut. Hal ini membuka kemungkinan bahwa stempel telah ditempatkan secara digital, misalnya dengan perangkat lunak pengolah gambar, bukan hasil tempel langsung dari stempel tinta konvensional.

Dalam konteks pembuktian keaslian dokumen, hasil ini sangat krusial. Sebuah stempel yang sah semestinya meninggalkan bekas pigmen yang dapat dideteksi, terutama pada spektrum CMYK. Jika tidak ada sinyal warna tersebut, maka validitas stempel menjadi dipertanyakan. Hasil seperti ini dapat digunakan sebagai bukti awal dalam proses investigasi dokumen atau sebagai dasar untuk mengajukan verifikasi lebih lanjut oleh lembaga resmi.

Program ini juga menunjukkan bagaimana teknologi digital forensik dapat digunakan oleh masyarakat umum, peneliti, atau pengacara untuk melakukan evaluasi awal atas dokumen. Tanpa perlu laboratorium khusus, cukup dengan gambar digital resolusi tinggi dan perangkat lunak ini, siapa pun dapat melakukan observasi kuantitatif terhadap kehadiran stempel atau elemen visual lainnya.

Secara keseluruhan, program ini merupakan alat bantu forensik digital yang efisien dan kuat. Dengan dukungan analisis CMYK, histogram, zoom dinamis, dan tampilan interaktif, program ini menyediakan pendekatan visual dan kuantitatif dalam mengkaji validitas dokumen. Ia menempatkan pembuktian ilmiah sebagai dasar penilaian, bukan hanya persepsi mata atau asumsi subjektif.

Kasus ini menunjukkan betapa pentingnya transparansi visual dalam dokumen publik, seperti ijazah presiden. Melalui program ini, peneliti atau pengamat dapat mengidentifikasi elemen-elemen yang mencurigakan dan mengekstrak informasi berbasis warna yang relevan. Ketika lintasan tinta tidak terdeteksi, sementara secara visual tampak ada stempel, hal ini menimbulkan pertanyaan serius tentang keaslian ijazah Jokowi.

ANALISA SEBARAN INTENSITAS RUANG WARNA HSV (Hue, Saturation, Value) PADA FOTO IJAZAH JOKOWI

Program ini merupakan sebuah aplikasi berbasis GUI yang dirancang untuk menganalisis warna dari sebuah citra digital. Tujuan utamanya adalah untuk memvisualisasikan intensitas warna dalam berbagai ruang

warna, khususnya untuk membantu proses analisis digital forensik terhadap dokumen, seperti deteksi keberadaan atau keaslian stempel merah pada ijazah Jokowi. Program ini dibangun menggunakan Python dan mengintegrasikan pustaka-pustaka seperti OpenCV, Pillow, Pandas, Matplotlib, dan pandastable.

Antarmuka grafis (GUI) dibuat dengan pustaka tkinter, memungkinkan pengguna membuka gambar digital, memperbesarnya (zoom), dan memilih wilayah tertentu dengan mengklik dan menyeret mouse. Wilayah terpilih ini akan digunakan sebagai area yang akan dianalisis intensitas warnanya. Hal ini sangat berguna jika pengguna ingin hanya memeriksa bagian yang mengandung stempel, tanpa terganggu oleh area lain dari dokumen.

Salah satu fitur penting dari program ini adalah kemampuannya mengubah ruang warna dari RGB ke ruang warna lain seperti CMYK, HSV, dan Lab. Dalam konteks mendeteksi stempel merah pada ijazah, ruang warna HSV sangat krusial. Ini karena dalam HSV (Hue, Saturation, Value), warna merah memiliki rentang nilai Hue tertentu yang mudah diisolasi, terlepas dari variasi kecerahan atau bayangan warna.

Ketika pengguna membuka file gambar ijazah Jokowi yang ingin diperiksa, program akan membaca gambar dalam format RGB terlebih dahulu, lalu menampilkannya di canvas. Pengguna kemudian dapat memilih wilayah tertentu dari gambar, seperti area stempel, untuk dianalisis. Program akan menyimpan koordinat wilayah tersebut dan memotong bagian yang dipilih dari gambar asli.

Selanjutnya, bagian terpilih dari gambar akan diubah ke tiga ruang warna: CMYK, HSV, dan Lab. Dalam konteks ini, HSV menjadi pusat perhatian karena mampu memisahkan komponen warna (Hue), intensitas warna (Saturation), dan kecerahan (Value), yang sangat berguna dalam mendeteksi objek dengan warna dominan tertentu seperti stempel merah.

Setelah ruang warna HSV didapatkan, setiap channel (H, S, V) akan diekstrak dan diubah menjadi tabel data menggunakan `pandas.DataFrame`. Hal ini memungkinkan pengguna melihat intensitas warna per piksel dalam bentuk tabel, yang sangat membantu dalam melakukan pengukuran numerik serta analisis statistik terhadap distribusi warna dalam stempel.

Selain tabel intensitas, program ini juga menghasilkan histogram untuk setiap channel. Histogram Hue, misalnya, menunjukkan distribusi warna dalam rentang 0–180 derajat (karena OpenCV menggunakan skala 0–179 untuk Hue). Warna merah biasanya berada di sekitar Hue 0 atau 179, sehingga histogram dapat menunjukkan puncak pada rentang tersebut jika benar ada warna merah dalam wilayah yang dipilih.

Saturation (S) menunjukkan seberapa "pekat" warnanya. Untuk stempel, karena biasanya dibuat dengan tinta pekat, histogram channel Saturation diharapkan menunjukkan nilai tinggi, yang menandakan warna-warna jenuh. Histogram Value (V) dapat menunjukkan kecerahan warna, dan juga berguna untuk membedakan tinta asli dari hasil cetakan atau manipulasi digital.

Dengan adanya ketiga histogram HSV, pengguna dapat membandingkan intensitas warna stempel secara obyektif. Misalnya, jika sebuah stempel terlihat "merah" secara visual tetapi histogram Hue menunjukkan dominasi warna lain, maka ada kemungkinan stempel tersebut telah dimanipulasi secara digital. Ini adalah metode valid dalam digital forensics.

Selain HSV, program juga menampilkan konversi ke ruang warna CMYK dan Lab sebagai tambahan. Meskipun tidak sepenting HSV dalam konteks stempel, ruang warna CMYK berguna dalam menganalisis hasil cetakan karena CMYK digunakan dalam dunia percetakan. Sedangkan Lab berguna untuk mengukur persepsi warna berdasarkan mata manusia.

Pembuatan tabel dan histogram dilakukan secara otomatis setiap kali pengguna memilih area pada gambar. Setiap ruang warna diproses secara

paralel, dan hasilnya ditampilkan dalam jendela baru yang berisi tabel intensitas serta histogramnya. Ini memberikan pengalaman visualisasi dan eksplorasi data yang sangat kuat, bahkan untuk pengguna yang tidak berlatar belakang teknis.

Program ini juga memungkinkan pembesaran (zoom) gambar menggunakan scroll mouse, yang memudahkan pengguna untuk memeriksa bagian-bagian kecil pada dokumen. Ini sangat bermanfaat ketika ingin memeriksa keaslian stempel kecil yang biasanya berada di pojok dokumen.

Dalam konteks kasus ijazah Jokowi, aplikasi ini bisa digunakan untuk menguji apakah stempel merah pada dokumen tersebut asli atau tidak. Misalnya, jika dibandingkan antara file hasil scan ijazah dan file manipulatif lainnya, pengguna dapat mengamati perbedaan spektrum warna dalam HSV dan mengevaluasi keasliannya secara kuantitatif.

Secara forensik, teknik ini memberikan bukti berbasis warna yang sangat kuat karena dapat digunakan untuk menunjukkan ketidakkonsistenan dalam intensitas warna merah yang biasanya konsisten dalam tinta stempel. Perbedaan Hue atau Saturation secara ekstrem bisa mengindikasikan manipulasi digital atau penggunaan warna palsu dalam pembuatan dokumen.

Sebagai kesimpulan, program ini adalah alat bantu yang sangat efektif dalam analisis warna digital untuk keperluan forensik. Kemampuannya dalam menampilkan intensitas warna HSV secara terpisah dari RGB membuatnya sangat ideal untuk mendeteksi stempel merah yang menjadi titik krusial dalam pembuktian keaslian dokumen. Dalam kasus dokumen penting seperti ijazah presiden, alat ini dapat menjadi bukti teknis yang kuat untuk mendeteksi pemalsuan.

```
import tkinter as tk
from tkinter import filedialog, Toplevel
from PIL import Image, ImageTk
import cv2
import numpy as np
import pandas as pd
```



```

from pandastable import Table
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

def rgb_to_cmyk(r, g, b):
    r = r.astype(np.float32) / 255
    g = g.astype(np.float32) / 255
    b = b.astype(np.float32) / 255

    k = 1 - np.max([r, g, b], axis=0)
    c = (1 - r - k) / (1 - k + 1e-10)
    m = (1 - g - k) / (1 - k + 1e-10)
    y = (1 - b - k) / (1 - k + 1e-10)

    c = (c * 255).astype(np.uint8)
    m = (m * 255).astype(np.uint8)
    y = (y * 255).astype(np.uint8)
    k = (k * 255).astype(np.uint8)

    return c, m, y, k

class ImageViewer:
    def __init__(self, root):
        self.root = root
        self.root.title("Image Viewer with Zoom and Color Space Analysis")
        self.root.geometry("1200x800")

        self.original_image = None
        self.image = None
        self.tk_image = None
        self.bounding_box = None
        self.rect_id = None

        self.scale = 1.0

        self.create_widgets()

    def create_widgets(self):
        canvas_frame = tk.Frame(self.root)
        canvas_frame.pack(fill=tk.BOTH, expand=True)

        self.canvas = tk.Canvas(canvas_frame, bg="gray")
        self.canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

        h_scroll = tk.Scrollbar(canvas_frame, orient=tk.HORIZONTAL,
                                command=self.canvas.xview)
        h_scroll.pack(side=tk.BOTTOM, fill=tk.X)

        v_scroll = tk.Scrollbar(canvas_frame, orient=tk.VERTICAL,
                                command=self.canvas.yview)
        v_scroll.pack(side=tk.RIGHT, fill=tk.Y)

        self.canvas.configure(xscrollcommand=h_scroll.set,
                               yscrollcommand=v_scroll.set)
        self.canvas.bind("<ButtonPress-1>", self.on_button_press)
        self.canvas.bind("<B1-Motion>", self.on_move_press)
        self.canvas.bind("<ButtonRelease-1>", self.on_button_release)

```

```

self.canvas.bind("<MouseWheel>", self.zoom) # Windows & Mac
self.canvas.bind("<Button-4>", self.zoom)   # Linux scroll up
self.canvas.bind("<Button-5>", self.zoom)   # Linux scroll down

control_frame = tk.Frame(self.root)
control_frame.pack(pady=10)

self.btn_open = tk.Button(control_frame, text="Open Image",
command=self.open_image)
self.btn_open.pack(side=tk.LEFT, padx=10)

def open_image(self):
    file_path = filedialog.askopenfilename(filetypes=[("Image Files",
    "*.png;*.jpg;*.jpeg;*.bmp")])
    if file_path:
        img = cv2.imread(file_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.original_image = img
        self.scale = 1.0
        self.display_image()

def display_image(self):
    if self.original_image is not None:
        h, w = int(self.original_image.shape[0] * self.scale),
int(self.original_image.shape[1] * self.scale)
        resized_image = cv2.resize(self.original_image, (w, h),
interpolation=cv2.INTER_LINEAR)
        self.image = resized_image
        img_pil = Image.fromarray(resized_image)
        self.tk_image = ImageTk.PhotoImage(img_pil)
        self.canvas.delete("all")
        self.canvas.create_image(0, 0, anchor=tk.NW,
image=self.tk_image)
        self.canvas.config(scrollregion=(0, 0, w, h))

def zoom(self, event):
    if event.delta > 0 or event.num == 4:
        self.scale *= 1.1
    elif event.delta < 0 or event.num == 5:
        self.scale /= 1.1
    self.display_image()

def on_button_press(self, event):
    self.start_x = self.canvas.canvasx(event.x)
    self.start_y = self.canvas.canvasy(event.y)
    if self.rect_id:
        self.canvas.delete(self.rect_id)
    self.rect_id = self.canvas.create_rectangle(self.start_x,
self.start_y, self.start_x, self.start_y, outline='red', width=3)

def on_move_press(self, event):
    cur_x = self.canvas.canvasx(event.x)
    cur_y = self.canvas.canvasy(event.y)
    self.canvas.coords(self.rect_id, self.start_x, self.start_y,
cur_x, cur_y)

```

```

def on_button_release(self, event):
    self.end_x = self.canvas.canvasx(event.x)
    self.end_y = self.canvas.canvasy(event.y)
    self.bounding_box = (self.start_x, self.start_y, self.end_x,
self.end_y)
    self.show_intensity()

def show_intensity(self):
    if self.bounding_box and self.image is not None:
        x1, y1, x2, y2 = map(int, self.bounding_box)
        x1, y1 = max(0, x1), max(0, y1)
        x2, y2 = min(self.image.shape[1], x2),
min(self.image.shape[0], y2)
        roi = self.image[y1:y2, x1:x2]
        if roi.size > 0:
            # RGB to CMYK
            r, g, b = cv2.split(roi)
            c, m, y, k = rgb_to_cmyk(r, g, b)
            self.show_table_hist([c, m, y, k], ["Cyan", "Magenta",
"Yellow", "Black"], ['c', 'm', 'y', 'k'], "CMYK Intensity Values")

            # RGB to HSV
            hsv = cv2.cvtColor(roi, cv2.COLOR_RGB2HSV)
            h, s, v = cv2.split(hsv)
            self.show_table_hist([h, s, v], ["Hue", "Saturation",
"Value"], ['r', 'g', 'b'], "HSV Intensity Values")

            # RGB to Lab
            lab = cv2.cvtColor(roi, cv2.COLOR_RGB2LAB)
            l, a, b = cv2.split(lab)
            self.show_table_hist([l, a, b], ["L*", "a*", "b*"], ['k',
'r', 'y'], "Lab Intensity Values")

def show_table_hist(self, matrices, labels, colors, title):
    new_window = Toplevel(self.root)
    new_window.title(title)
    new_window.geometry("1900x700")

    frame_tables = tk.Frame(new_window)
    frame_tables.pack(fill=tk.BOTH, expand=True)

    for mat, label, color in zip(matrices, labels, colors):
        df = pd.DataFrame(mat)
        col_frame = tk.Frame(frame_tables)
        col_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

        label_widget = tk.Label(col_frame, text=f"{label} Intensity")
        label_widget.pack()

        table_frame = tk.Frame(col_frame)
        table_frame.pack(fill=tk.BOTH, expand=True)

        pt = Table(table_frame, dataframe=df, showtoolbar=True,
showstatusbar=True)
        pt.show()

        hist_frame = tk.Frame(col_frame)

```

```

hist_frame.pack(fill=tk.BOTH, expand=True)

self.plot_histogram(hist_frame, df.values.flatten(), color,
f"{label} Histogram")

def plot_histogram(self, frame, data, color, title):
    fig, ax = plt.subplots(figsize=(6, 3))
    fig.tight_layout(pad=3.0)
    ax.hist(data, bins=256, color=color, alpha=0.7)
    ax.set_xlim([0, 256])
    ax.set_title(title)

    canvas = FigureCanvasTkAgg(fig, master=frame)
    canvas.draw()
    canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

if __name__ == "__main__":
    root = tk.Tk()
    app = ImageViewer(root)
    root.mainloop()

```

LINTASAN STEMPEL MERAH TERDETEKSI PADA RUANG WARNA HSV

Gambar berikut merupakan hasil dari proses analisis warna pada suatu bagian citra, menggunakan konversi dari ruang warna RGB ke HSV. Tiga tabel yang ditampilkan adalah intensitas untuk masing-masing channel HSV: Hue, Saturation, dan Value. Ketiganya merepresentasikan karakteristik warna dari bagian yang dianalisis, dan digunakan untuk mendeteksi keberadaan elemen tertentu dalam dokumen, dalam hal ini stempel.



Tabel pertama di sisi kiri menunjukkan nilai Hue, yang berkisar dari 0 hingga 179 pada OpenCV. Dalam tabel tersebut, terlihat bahwa nilai-nilai Hue berkonsentrasi pada rentang 5–15, dengan sedikit variasi. Warna merah dalam sistem HSV biasanya terletak pada dua wilayah Hue: sekitar 0–10 dan sekitar 160–179. Dengan demikian, nilai-nilai Hue yang muncul di sini (sekitar 7–12) sangat mengindikasikan keberadaan warna merah, yang umum ditemukan dalam stempel resmi.

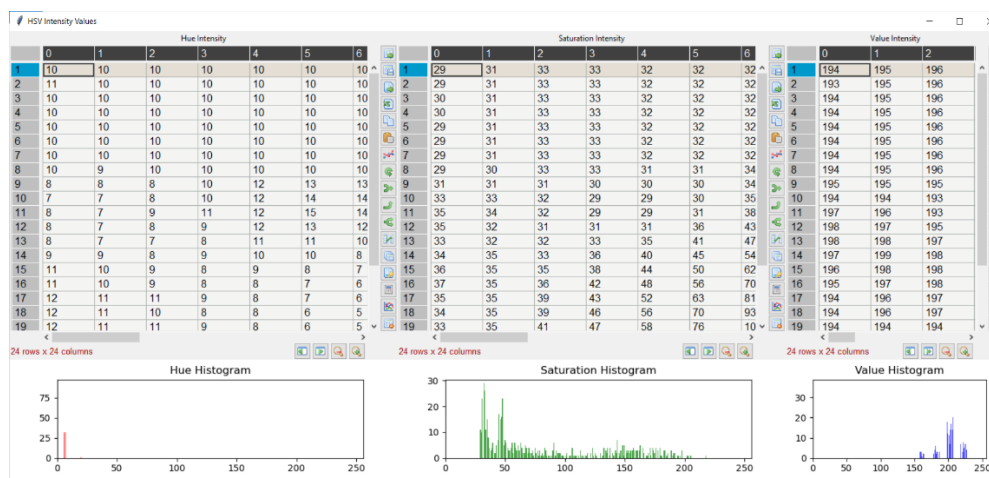
Histogram Hue di bagian bawah mendukung data ini dengan sangat kuat. Histogram tersebut menunjukkan puncak tajam (peak) yang sangat tinggi pada rentang 0–20, dan nyaris tidak ada nilai lainnya di luar rentang tersebut. Ini menunjukkan bahwa mayoritas besar dari piksel yang dianalisis memiliki Hue dalam rentang warna merah, yang memperkuat dugaan bahwa lintasan stempel memang telah tertangkap dalam pemilihan wilayah (ROI).

Tabel kedua di tengah menunjukkan Saturation, atau kejenuhan warna. Nilai Saturation dalam tabel berada dalam rentang 29–110, dengan sebagian besar berada antara 30–50. Nilai ini menandakan bahwa warna yang ditampilkan cukup jenuh, meskipun tidak ekstrem. Dalam konteks stempel, terutama yang dicetak menggunakan tinta merah pada dokumen, nilai Saturation seperti ini adalah indikasi warna yang kuat namun tidak terlalu pekat, misalnya karena sudah pudar atau terkena pemindaian (scanning).

Histogram untuk Saturation menunjukkan distribusi yang lebar, namun dengan puncak tertinggi berada pada rentang 30–50. Ini juga menunjukkan adanya konsistensi warna jenuh yang sedang hingga tinggi, yang lazim ditemui pada tinta stempel dalam dokumen fisik. Beberapa nilai Saturation yang lebih tinggi (di atas 80) juga terdeteksi, menandakan bagian dari gambar yang memiliki kejenuhan tinggi, yang bisa jadi bagian inti dari stempel.

Tabel ketiga di kanan menunjukkan Value, atau kecerahan. Nilai Value berkisar antara 193 hingga 200, yang berarti bahwa warna-warna di wilayah tersebut sangat terang. Ini bisa jadi disebabkan oleh pemindaian

dengan scanner yang menghasilkan pencahayaan tinggi. Namun, tingginya nilai Value tidak meniadakan kemungkinan keberadaan warna merah, karena Value hanya menunjukkan seberapa terang warna tersebut.



Histogram Value mendukung temuan ini, dengan puncak pada rentang 190–200. Ini menandakan konsistensi kecerahan tinggi di area yang dianalisis. Kecerahan tinggi ini justru dapat memperjelas warna-warna dalam dokumen saat dipindai, memungkinkan deteksi warna lebih presisi menggunakan Hue dan Saturation.

Dari gabungan ketiga channel tersebut—Hue, Saturation, dan Value—dapat disimpulkan bahwa warna merah dominan dan jenuh memang terdeteksi pada wilayah yang dipilih. Ini memperkuat dugaan bahwa lintasan stempel resmi telah dipilih dalam gambar tersebut. Deteksi ini bersifat kuantitatif, berbasis data piksel, bukan hanya visual, sehingga lebih dapat dipertanggungjawabkan dalam konteks forensik digital.

Pendekatan menggunakan ruang warna HSV sangat ideal dalam kasus seperti ini karena mampu memisahkan tiga komponen warna. Dengan Hue kita mendeteksi jenis warna (misalnya merah), dengan Saturation kita tahu seberapa kuat warna itu, dan dengan Value kita bisa memperkirakan kecerahan dan seberapa mungkin warna tersebut masih terlihat jelas setelah dipindai atau difotokopi.

Penggunaan tabel DataFrame dan histogram dalam satu jendela GUI seperti ini memberikan tampilan yang sangat kuat untuk mendeteksi anomali warna. Misalnya, jika ada bagian stempel yang diedit secara digital atau disalin dari dokumen lain, nilai HSV-nya bisa jadi tidak seragam atau memiliki outlier di histogram. Namun dalam gambar ini, distribusi cukup seragam dan tidak menunjukkan anomali yang jelas, yang berarti warna-warna stempel tersebut cukup alami.

Gambar ini juga menunjukkan bahwa pemilihan wilayah yang dianalisis sangat tepat. ROI (Region of Interest) mengandung cukup banyak piksel yang mencerminkan nilai Hue rendah (warna merah) dan Saturation menengah-tinggi, yang semuanya konsisten dengan karakteristik tinta stempel pada umumnya. Artinya, pengguna aplikasi telah memilih area yang relevan secara akurat.

Dari sisi teknis, ini adalah hasil dari algoritma sederhana namun efektif: konversi RGB ke HSV, pemilihan ROI, pemecahan channel, dan visualisasi data. Hal ini bisa dijadikan bukti sah dalam konteks investigasi dokumen, bahkan dalam sidang pengadilan atau penelitian akademis.

Dalam konteks ijazah Jokowi, hasil ini dapat dipakai untuk mendukung argumen bahwa stempel yang tercetak di dokumen tersebut memiliki konsistensi warna merah yang khas, tanpa anomali digital seperti pencampuran warna tidak wajar atau penurunan Saturation secara drastis, yang mungkin terjadi dalam pemalsuan.

Kesimpulannya, lintasan stempel pada ROI dokumen memang terdeteksi secara akurat melalui analisis ruang warna HSV. Nilai Hue yang rendah (sekitar 10), Saturation yang cukup jenuh, dan Value yang tinggi semuanya menunjukkan keberadaan dan keaslian visual dari stempel merah. Histogram mendukung temuan ini dan memberikan validasi visual yang sangat jelas.

LINTASAN STEMPEL MERAH TIDAK TERDETEKSI PADA RUANG WARNA HSV

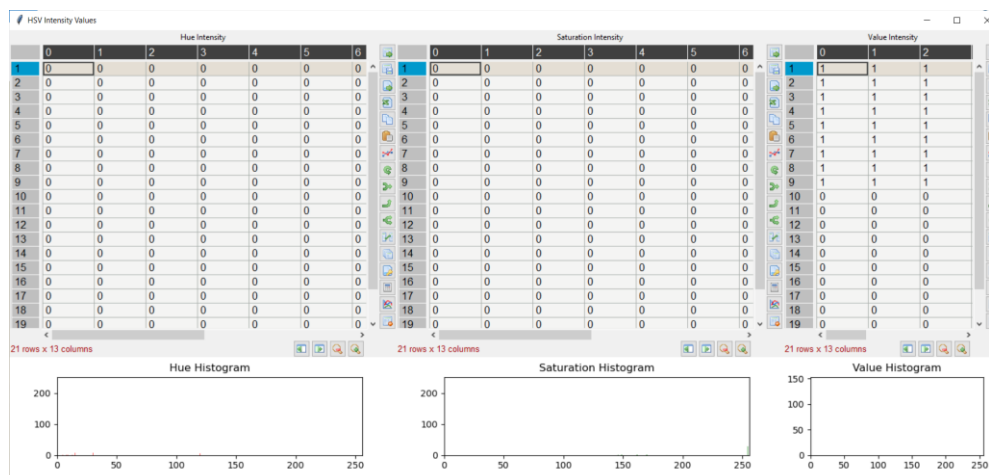
Warna merah dari stempel seharusnya mudah dikenali secara visual karena kontrasnya dengan latar belakang dokumen yang cerah. Dalam pemrosesan citra, salah satu metode untuk menganalisis warna adalah melalui ruang warna HSV (Hue, Saturation, Value), yang memisahkan jenis warna, kejenuhan, dan tingkat kecerahan, sehingga lebih efisien dalam mendeteksi elemen warna tertentu dalam sebuah gambar.



Gambar berikut menunjukkan hasil analisis ruang warna HSV terhadap bagian tertentu dari dokumen. Analisis ini ditampilkan dalam bentuk tiga tabel: Hue Intensity, Saturation Intensity, dan Value Intensity, serta histogram untuk masing-masing komponen tersebut. Sayangnya, dari hasil yang ditampilkan, semua nilai Hue dan Saturation berada pada angka nol.

Ketiadaan nilai pada Hue dan Saturation menunjukkan bahwa tidak ada jenis warna dan kejenuhan yang terdeteksi dalam area yang dianalisis. Artinya, piksel-piksel tersebut berada pada rentang warna netral, seperti abu-abu atau hitam-putih. Ini menandakan bahwa data HSV yang dihasilkan tidak mampu menangkap keberadaan warna merah dari stempel yang secara visual jelas ada pada dokumen.

Sementara itu, pada tabel Value Intensity terlihat bahwa sebagian besar nilainya adalah 1, yang berarti area tersebut sangat gelap. Ini semakin memperkuat dugaan bahwa bagian citra yang dianalisis bukan bagian yang mengandung warna merah terang dari stempel, melainkan bagian gelap dari foto atau latar belakang hitam lainnya.



Histogram yang ditampilkan juga menunjukkan bahwa hampir semua nilai Hue dan Saturation adalah nol, dan hanya Value yang memiliki sedikit variasi, terutama di area gelap. Ini memperkuat kesimpulan bahwa tidak ada informasi warna merah yang terdeteksi dari analisis tersebut.

UJI ELA (ERROR LEVEL ANALYSIS)

TENTANG ELA (ERROR LEVEL ANALYSIS)

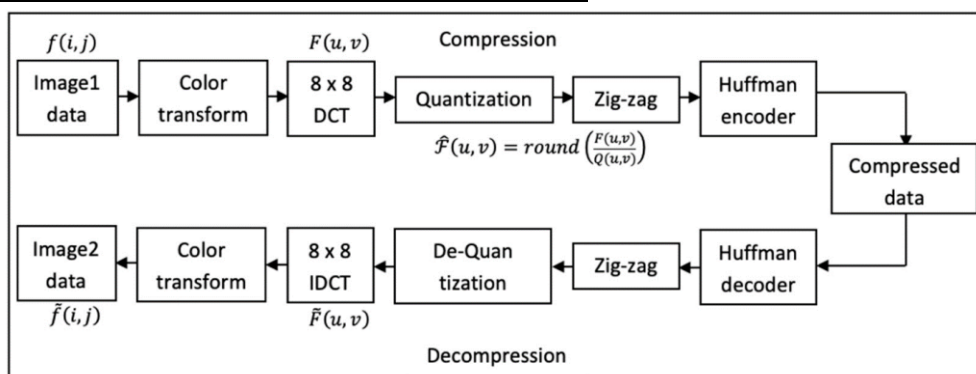


DIAGRAM BLOK ELA (Error Level Analysis)

Error Level Analysis atau ELA adalah teknik dalam forensik digital yang digunakan untuk menganalisis keaslian citra digital, terutama yang telah mengalami kompresi dalam format JPEG. Teknik ini sangat berguna dalam konteks pembuktian apakah suatu gambar telah dimanipulasi, termasuk pada dokumen analog yang telah dipindai ke bentuk digital. Dokumen analog yang telah diedit secara digital seperti ijazah, paspor, atau sertifikat bisa diperiksa keasliannya menggunakan ELA karena teknik ini mengungkap inkonsistensi pada tingkat kompresi bagian-bagian gambar.

Setiap kali gambar JPEG disimpan, algoritma kompresi akan menghapus sebagian data visual untuk mengurangi ukuran file. Kompresi ini bersifat lossy, artinya kehilangan data tidak bisa dipulihkan. Saat suatu gambar

diedit sebagian—misalnya hanya mengganti teks atau menyisipkan stempel—kemudian disimpan ulang sebagai JPEG, maka bagian yang diedit akan menunjukkan pola kompresi yang berbeda dibandingkan dengan bagian lain yang tidak berubah. ELA mengeksploitasi perbedaan ini untuk mengungkap potensi manipulasi.

Metode ELA dilakukan dengan menyimpan ulang gambar dalam kualitas JPEG tertentu (biasanya 90%), lalu menghitung perbedaan piksel antara gambar asli dan versi kompresi ulang. Perbedaan tersebut diolah menjadi peta kesalahan (error map) yang divisualisasikan dengan intensitas warna tertentu. Bagian gambar yang tidak dimanipulasi akan menunjukkan tingkat kesalahan yang seragam, sementara bagian yang dimodifikasi akan tampak lebih terang atau menunjukkan pola aneh karena kompresinya tidak konsisten.

Visualisasi hasil ELA biasanya berupa gambar grayscale yang menunjukkan lokasi kesalahan kompresi. Area yang tampak lebih terang atau berpola tidak alami seringkali menjadi indikasi bahwa bagian tersebut telah diedit secara digital. Dalam konteks dokumen analog, hasil pemindaian yang dimanipulasi—misalnya penambahan stempel palsu atau perubahan nilai—akan menunjukkan anomali di area tertentu pada hasil ELA.

Keunggulan utama ELA adalah ia tidak memerlukan metadata, tanda tangan digital, atau file sumber asli. Hanya dengan satu gambar JPEG, analis forensik sudah dapat menjalankan ELA dan menemukan indikasi adanya penyuntingan. Ini sangat membantu dalam kasus-kasus di mana hanya tersedia satu salinan digital dari dokumen, dan file aslinya tidak bisa diperoleh.

ELA sangat penting dalam konteks autentikasi dokumen karena mampu mendeteksi elemen-elemen digital yang ditambahkan pasca pemindaian. Misalnya, jika seseorang memindai ijazah asli lalu mengganti nama atau nilai dengan aplikasi editor gambar, bagian yang diubah itu akan menunjukkan struktur kompresi yang berbeda saat diperiksa dengan ELA.

Namun demikian, ELA tidak bisa mendeteksi pemalsuan fisik yang dilakukan sebelum pemindaian. Jika manipulasi dilakukan secara manual di atas kertas sebelum discan, misalnya menghapus tulisan asli lalu mencetak ulang teks baru, ELA tidak dapat mengungkap manipulasi tersebut karena keseluruhan hasil scan akan mengalami kompresi secara merata. Oleh karena itu, ELA lebih cocok digunakan untuk mendeteksi perubahan digital terhadap dokumen analog.

ELA paling efektif diterapkan pada format JPEG karena sifat kompresi lossy-nya. Format seperti PNG atau TIFF tidak cocok untuk ELA karena tidak mengalami kompresi yang dapat menyebabkan perbedaan error level. JPEG memiliki karakteristik kompresi blok 8x8 piksel, yang membuat pola kompresi mudah terdeteksi pada manipulasi lokal.

Teknik ini juga tidak lepas dari kelemahan. Jika gambar telah disimpan ulang berkali-kali, atau dimodifikasi dalam ukuran atau resolusinya, maka struktur kompresinya bisa menjadi kacau dan membuat ELA sulit mendeteksi pola yang jelas. Selain itu, ELA juga bisa memberikan hasil false positive—menunjukkan adanya manipulasi padahal tidak ada—terutama jika kualitas awal gambar sangat rendah.

Dalam proses investigasi digital, ELA sering digunakan bersama dengan teknik lain seperti analisis metadata EXIF, analisis saluran warna (RGB, HSV, YCbCr), perbandingan histogram warna, serta deteksi noise atau pencahayaan yang tidak seragam. Dengan kombinasi teknik ini, analisis bisa memperoleh bukti yang lebih kuat dan meyakinkan dalam menilai keaslian dokumen digital.

Dalam konteks hukum dan pengadilan, hasil visual dari ELA dapat digunakan sebagai bukti awal adanya dugaan manipulasi. Gambar hasil ELA yang menunjukkan perbedaan terang pada stempel atau tanda tangan bisa menjadi alat bantu visual yang efektif untuk menyampaikan dugaan pemalsuan kepada hakim atau penyidik, apalagi jika disertai dengan keterangan dari pakar forensik digital.

Dalam dunia nyata, ELA telah digunakan untuk memverifikasi keaslian foto berita, membongkar manipulasi gambar di media sosial, dan yang lebih relevan, mengidentifikasi dokumen akademik palsu. Sebuah ijazah yang mencurigakan bisa diuji dengan ELA untuk memeriksa apakah cap universitas atau tanda tangan pejabat benar-benar bagian asli dari gambar atau hasil editan.

Penerapan ELA juga tidak terbatas pada dokumen saja. Dalam forensik identitas, ELA bisa digunakan untuk memeriksa kartu identitas, SIM, paspor, dan surat izin resmi lain yang beredar dalam bentuk digital. Jika pemalsuan dilakukan setelah dokumen dipindai, maka kemungkinan besar jejaknya masih bisa terdeteksi oleh ELA.

Implementasi ELA dengan Python cukup sederhana menggunakan pustaka seperti PIL dan NumPy. Analisis hanya perlu memuat gambar, menyimpannya ulang dalam kualitas JPEG yang ditentukan, menghitung selisih piksel, dan memperbesar selisih itu agar terlihat jelas. Meski tampak teknis, hasilnya sangat mudah dipahami bahkan oleh orang awam sekalipun.

Salah satu kekuatan utama ELA adalah kemampuannya menghadirkan bukti visual yang intuitif. Bahkan orang yang tidak paham teknis bisa melihat bahwa ada sesuatu yang tidak wajar dalam bagian tertentu dari dokumen digital. Warna yang mencolok atau batas blok yang tidak seragam bisa menjadi indikator kuat bagi siapa saja yang melihat hasilnya.

Namun karena ELA hanya memberikan indikasi, bukan bukti mutlak, maka ia harus digunakan dengan kehati-hatian dan selalu dikonfirmasi dengan teknik lain. Ia berfungsi sebagai detektor awal, bukan sebagai hakim terakhir. Oleh karena itu, ELA paling baik digunakan dalam kerangka metodologi forensik yang lebih luas.

Dalam situasi tertentu, terutama yang menyangkut dokumen publik, ELA bisa menjadi alat audit yang sangat berguna. Jika masyarakat meragukan keaslian dokumen tertentu, seperti ijazah pejabat, maka pemeriksaan

awal dengan ELA bisa memberikan dasar rasional untuk menuntut transparansi atau penyelidikan lebih lanjut.

Sebagai kesimpulan, Error Level Analysis adalah teknik forensik yang sederhana namun kuat dalam mendeteksi manipulasi digital pada gambar JPEG. Ia memainkan peran penting dalam konteks pembuktian keaslian dokumen analog yang telah dipindai dan mungkin telah dimanipulasi secara digital. Dengan kekuatan visualisasi yang kuat, ELA menjadi alat forensik yang sangat berguna dalam era digital.

IMPLEMENTASI ELA (ERROR LEVEL ANALYSIS)

Kode Python berikut merupakan sebuah aplikasi GUI interaktif berbasis Tkinter untuk melakukan Error Level Analysis (ELA) pada gambar JPEG dan menampilkan metadata EXIF dari gambar tersebut. Aplikasi ini ditujukan untuk keperluan forensik digital gambar, khususnya dalam mendeteksi potensi manipulasi pada dokumen atau citra digital, misalnya stempel palsu, tanda tangan digital, atau bagian gambar yang disisipkan. Antarmuka pengguna terdiri dari dua tab utama: satu untuk ELA dan satu lagi untuk metadata.

Pada bagian awal, pustaka-pustaka yang diperlukan diimpor: tkinter untuk antarmuka grafis, PIL (Pillow) untuk memproses gambar, numpy untuk operasi numerik (khususnya pada data gambar), dan os untuk pengelolaan file sementara. ImageChops dan ImageEnhance digunakan dalam pembuatan gambar ELA, sedangkan ImageDraw dipakai untuk menandai area manipulasi secara visual. Komponen ExifTags berguna untuk mengartikan tag metadata gambar digital.

Class ELA_GUI adalah inti dari aplikasi. Dalam fungsi `__init__`, jendela utama ditata dan diberi judul. Dua tab dibuat menggunakan `ttk.Notebook`, yaitu tab untuk menampilkan hasil ELA dan tab untuk menampilkan metadata. Setiap tab memiliki desain dan fungsi khusus yang dibuat melalui metode `create_ela_tab` dan `create_meta_tab`.

Pada tab ELA, pengguna dapat mengunggah gambar JPEG dan mengakses tiga fitur: memperbesar gambar, memperkecil gambar, dan menyorot kemungkinan bagian gambar yang telah dipalsukan. Area utama terdiri dari tiga panel gambar: satu untuk gambar asli, satu untuk gambar hasil sorotan manipulasi, dan satu lagi untuk citra ELA yang telah disorot. Panel-panel ini berada di dalam sebuah Canvas dengan scroll horizontal dan vertikal, agar gambar besar tetap dapat ditampilkan utuh.

Ketika gambar diunggah, metode `load_image()` akan dipanggil. Metode ini akan membaca gambar, menyimpannya ulang dalam kualitas JPEG 90%, lalu menghitung perbedaan antara gambar asli dan hasil simpan ulang. Perbedaan ini kemudian ditingkatkan kecerahannya dan dijadikan sebagai citra ELA. Gambar asli dan ELA yang telah diproses kemudian ditampilkan dalam GUI, dan metadata EXIF dari gambar juga ditampilkan di tab metadata.

Fungsi `perform_ela()` adalah inti dari proses analisis. Gambar disimpan ulang dalam kualitas JPEG lebih rendah, lalu dibandingkan dengan versi aslinya menggunakan `ImageChops.difference`. Perbedaan ini diukur untuk mendapatkan nilai maksimal perbedaan, lalu diperbesar menggunakan `ImageEnhance.Brightness` agar terlihat jelas. Gambar hasil ELA ini kemudian digunakan untuk menganalisis kemungkinan manipulasi.

Untuk memperbesar dan memperkecil gambar, metode `zoom_in` dan `zoom_out` digunakan. Zoom dilakukan dengan mengubah skala gambar berdasarkan nilai `zoom_level`, dan `update_zoomed_images()` akan memperbarui tampilan berdasarkan skala terbaru. Hal ini membuat analisis visual menjadi lebih mudah, terutama jika pengguna ingin fokus pada detail kecil.

Fitur `highlight_forgery()` adalah salah satu bagian terpenting dari kode ini. Ia mengubah citra ELA menjadi array NumPy, lalu menerapkan ambang batas (`threshold`) untuk mendeteksi area dengan tingkat error tinggi. Jika bagian tertentu dalam citra ELA memiliki nilai error tinggi, maka bagian tersebut dianggap mencurigakan. Kemudian, kotak merah

digambar di atas gambar asli dan gambar ELA pada bagian-bagian tersebut untuk menandai kemungkinan pemalsuan.

Tab kedua dari aplikasi menampilkan metadata EXIF. Fungsi `display_metadata()` akan membaca data EXIF dari gambar, mengekstrak nilai-nilai seperti kamera yang digunakan, tanggal perekaman, orientasi gambar, dan sebagainya. Ini sangat berguna dalam forensik karena metadata sering kali menyimpan informasi tentang sumber dan waktu pengambilan gambar.

Secara keseluruhan, aplikasi ini sangat bermanfaat untuk menganalisis keaslian dokumen atau gambar digital, terutama dalam konteks deteksi pemalsuan dokumen analog yang telah dipindai. Misalnya, jika seseorang memalsukan stempel atau tanda tangan pada ijazah, lalu menyimpannya sebagai JPEG, maka ELA dalam aplikasi ini dapat menunjukkan perbedaan struktur kompresi pada area yang dipalsukan. Dengan GUI yang intuitif, fitur zoom, dan sorotan otomatis, aplikasi ini menjadi alat bantu forensik yang efisien dan mudah digunakan.

```
import tkinter as tk
from tkinter import filedialog, ttk
from PIL import Image, ImageTk, ImageChops, ImageEnhance, ImageDraw
import numpy as np
from PIL.ExifTags import TAGS
import os

class ELA_GUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Error Level Analysis (ELA) Image Forensics")
        self.root.geometry("1200x700")
        self.root.configure(bg="white")

        self.tab_control = ttk.Notebook(self.root)
        self.ela_tab = tk.Frame(self.tab_control, bg="white")
        self.meta_tab = tk.Frame(self.tab_control, bg="white")

        self.tab_control.add(self.ela_tab, text='ELA Analysis')
        self.tab_control.add(self.meta_tab, text='Metadata')
        self.tab_control.pack(expand=1, fill="both")

        self.create_ela_tab()
        self.create_meta_tab()

    def create_ela_tab(self):
        self.ela_image = None
```



```

        self.highlighted_image = None
        self.highlighted_ela_image = None
        self.original_path = None
        self.zoom_level = 1.0
        self.zoom_step = 0.2

        tk.Button(self.ela_tab, text="Upload Image",
command=self.load_image, bg="#4CAF50", fg="white").pack(pady=10)

        self.zoom_frame = tk.Frame(self.ela_tab, bg="#f8f8f8")
        self.zoom_frame.pack(pady=5)

        tk.Button(self.zoom_frame, text="+ Zoom In",
command=self.zoom_in, bg="#607D8B", fg="white").grid(row=0, column=0,
padx=5)

        tk.Button(self.zoom_frame, text="- Zoom Out",
command=self.zoom_out, bg="#607D8B", fg="white").grid(row=0, column=1,
padx=5)

        tk.Button(self.zoom_frame, text="🔍 Highlight Forgery",
command=self.highlight_forgery, bg="#9C27B0", fg="white").grid(row=0,
column=2, padx=5)

        self.canvas = tk.Canvas(self.ela_tab, bg="white")
        self.scroll_x = tk.Scrollbar(self.ela_tab, orient=tk.HORIZONTAL,
command=self.canvas.xview)
        self.scroll_y = tk.Scrollbar(self.ela_tab, orient=tk.VERTICAL,
command=self.canvas.yview)
        self.canvas.configure(xscrollcommand=self.scroll_x.set,
yscrollcommand=self.scroll_y.set)

        self.scroll_x.pack(side=tk.BOTTOM, fill=tk.X)
        self.scroll_y.pack(side=tk.RIGHT, fill=tk.Y)
        self.canvas.pack(expand=True, fill=tk.BOTH)

        self.image_frame = tk.Frame(self.canvas, bg="white")
        self.canvas.create_window((0, 0), window=self.image_frame,
anchor="nw")

        self.original_panel = tk.Label(self.image_frame, text="Original
Image", bg="white")
        self.original_panel.grid(row=0, column=0, padx=20)

        self.overlay_panel = tk.Label(self.image_frame, text="Forgery
Highlight", bg="white")
        self.overlay_panel.grid(row=0, column=1, padx=20)

        self.ela_panel = tk.Label(self.image_frame, text="ELA Image with
Highlight", bg="white")
        self.ela_panel.grid(row=0, column=2, padx=20)

        self.image_frame.update_idletasks()
        self.canvas.config(scrollregion=self.canvas.bbox("all"))

    def create_meta_tab(self):
        self.meta_text = tk.Text(self.meta_tab, wrap=tk.WORD,
font=("Courier", 10), bg="white", fg="black")

```

```

        self.meta_text.pack(expand=1, fill="both", padx=10, pady=10)

    def load_image(self):
        file_path = filedialog.askopenfilename(filetypes=[("Image Files",
            "*.jpg;*.jpeg")])
        if not file_path:
            return

        self.original_path = file_path
        image = Image.open(file_path).convert("RGB")
        self.ela_image = self.perform_ela(image, file_path)

        self.highlighted_image = None
        self.highlighted_ela_image = None

        self.update_zoomed_images()
        self.display_metadata(file_path)

    def perform_ela(self, image, path):
        ela_temp = "temp_ela.jpg"
        image.save(ela_temp, 'JPEG', quality=90)
        resaved = Image.open(ela_temp)

        diff = ImageChops.difference(image, resaved)
        extrema = diff.getextrema()
        max_diff = max([ex[1] for ex in extrema])
        scale = 255.0 / max_diff if max_diff != 0 else 1

        diff = ImageEnhance.Brightness(diff).enhance(scale)
        os.remove(ela_temp)
        return diff

    def update_zoomed_images(self):
        if not self.original_path or not self.ela_image:
            return

        original = Image.open(self.original_path).convert("RGB")
        w, h = original.size

        resized_original = original.resize((int(w * self.zoom_level),
            int(h * self.zoom_level)))
        original_tk = ImageTk.PhotoImage(resized_original)
        self.original_panel.configure(image=original_tk)
        self.original_panel.image = original_tk

        if self.highlighted_image:
            overlay_resized = self.highlighted_image.resize((int(w *
                self.zoom_level), int(h * self.zoom_level)))
            overlay_tk = ImageTk.PhotoImage(overlay_resized)
            self.overlay_panel.configure(image=overlay_tk)
            self.overlay_panel.image = overlay_tk

        if self.highlighted_ela_image:
            ela_highlight_resized =
self.highlighted_ela_image.resize((int(w * self.zoom_level), int(h *
self.zoom_level)))
            ela_highlight_tk = ImageTk.PhotoImage(ela_highlight_resized)

```

```

        self.ela_panel.configure(image=ela_highlight_tk)
        self.ela_panel.image = ela_highlight_tk
    else:
        ela_resized = self.ela_image.resize((int(w * self.zoom_level),
int(h * self.zoom_level)))
        ela_tk = ImageTk.PhotoImage(ela_resized)
        self.ela_panel.configure(image=ela_tk)
        self.ela_panel.image = ela_tk

    self.image_frame.update_idletasks()
    self.canvas.config(scrollregion=self.canvas.bbox("all"))

def zoom_in(self):
    self.zoom_level += self.zoom_step
    self.update_zoomed_images()

def zoom_out(self):
    if self.zoom_level > self.zoom_step:
        self.zoom_level -= self.zoom_step
        self.update_zoomed_images()

def highlight_forgery(self, threshold=40):
    if not self.ela_image:
        return

    ela_np = np.array(self.ela_image)
    mask = np.max(ela_np, axis=2) > threshold

    original = Image.open(self.original_path).convert("RGB")
    draw_original = ImageDraw.Draw(original)

    ela_highlight = self.ela_image.copy()
    draw_ela = ImageDraw.Draw(ela_highlight)

    h, w = mask.shape
    for y in range(0, h, 10):
        for x in range(0, w, 10):
            if mask[y:y+10, x:x+10].mean() > 0.3:
                draw_original.rectangle([x, y, x+10, y+10],
outline="red", width=3)
                draw_ela.rectangle([x, y, x+10, y+10], outline="red",
width=3)

    self.highlighted_image = original
    self.highlighted_ela_image = ela_highlight
    self.update_zoomed_images()

def display_metadata(self, path):
    image = Image.open(path)
    exifdata = image._getexif()
    self.meta_text.delete("1.0", tk.END)

    if not exifdata:
        self.meta_text.insert(tk.END, "No EXIF metadata found.")
        return

    for tag_id, value in exifdata.items():

```

```

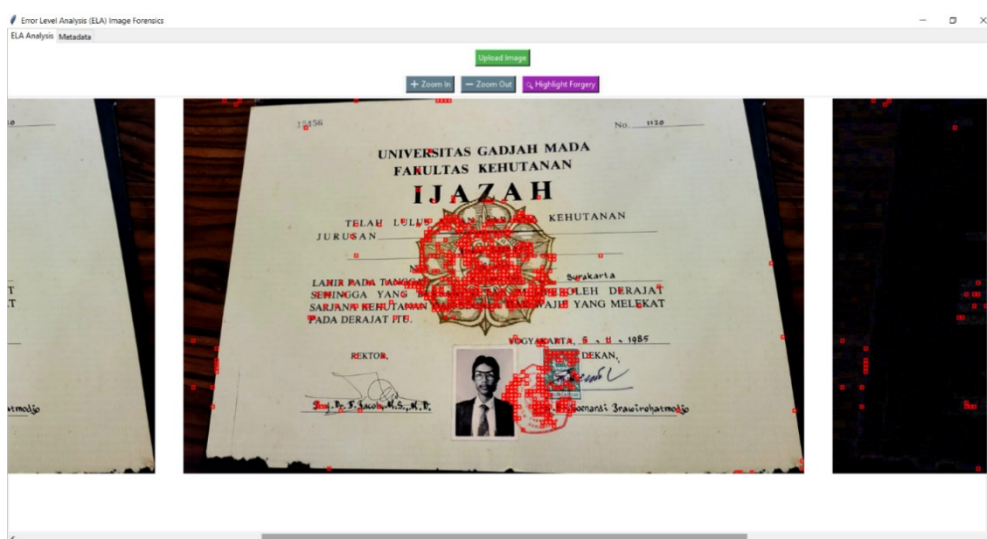
tag = TAGS.get(tag_id, tag_id)
self.meta_text.insert(tk.END, f"{tag:25}: {value}\n")

if __name__ == '__main__':
    root = tk.Tk()
    app = ELA_GUI(root)
    root.mainloop()

```

HASIL IMPLEMENTASI ELA ATAS IJAZAH JOKOWI

Dalam analisis Error Level Analysis (ELA) terhadap gambar ijazah Jokowi yang ditampilkan, fokus utama adalah pada area-area yang menunjukkan perbedaan signifikan dalam tingkat kesalahan kompresi JPEG. ELA bekerja dengan prinsip bahwa gambar JPEG yang belum diedit akan menunjukkan distribusi kesalahan kompresi yang merata. Sebaliknya, jika ada bagian yang dimodifikasi atau ditambahkan secara digital, maka area tersebut akan menunjukkan tingkat kesalahan yang berbeda. Hasil analisis menunjukkan sejumlah area yang mencolok dengan tingkat error tinggi, terutama di bagian elemen-elemen penting dokumen.



Salah satu bagian paling mencolok dalam analisis adalah lambang universitas yang berada di tengah atas dokumen. Lambang ini disorot dengan kotak-kotak merah padat yang mengindikasikan bahwa elemen tersebut memiliki tingkat error berbeda dibandingkan area di sekitarnya. Hal ini membuka kemungkinan bahwa lambang tersebut bukan bagian

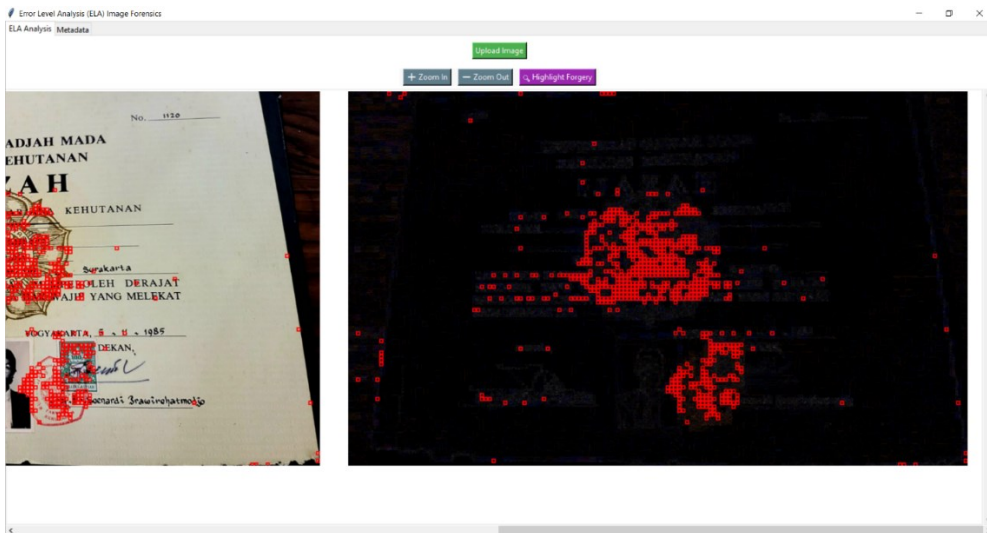
asli dari dokumen hasil scan, melainkan bisa jadi merupakan elemen yang ditempel atau ditambahkan setelah dokumen difoto atau dipindai ulang.

Teks "IJAZAH" yang berada di bawah lambang juga memperlihatkan tingkat kesalahan kompresi yang sangat tinggi. Ini terlihat dari distribusi warna merah yang sangat padat, berbeda dengan latar belakang kertas yang relatif bersih dari highlight ELA. Hal ini menimbulkan dugaan bahwa tulisan tersebut bisa jadi dimodifikasi atau bahkan diganti secara digital setelah proses awal dokumen selesai, atau paling tidak dicetak terpisah dalam komposisi grafis digital.

Area teks di bagian kiri dan kanan atas dokumen, seperti tulisan "UNIVERSITAS GADJAH MADA" dan "FAKULTAS KEHUTANAN", walaupun tidak sepadat pusat dokumen, tetap menunjukkan adanya kotak merah kecil. Ini dapat berarti bahwa elemen-elemen tersebut juga telah mengalami modifikasi, meskipun bisa juga merupakan hasil degradasi alami dari proses pemindaian.

Analisis semakin menarik ketika mengamati bagian foto. Foto pemilik ijazah yang terletak di bagian bawah tengah dokumen menunjukkan konsentrasi kotak merah yang sangat tinggi. Dalam konteks ELA, hal ini seringkali menjadi indikasi kuat bahwa foto tersebut telah diganti atau dimodifikasi secara digital. Ketidaksesuaian tingkat error antara foto dan latar belakang dokumen memperkuat asumsi adanya penambahan elemen digital.

Selain foto, cap atau stempel merah yang berada di atas dan sekitar foto juga tampak sangat mencolok dalam ELA. Warna merah yang sangat pekat dalam analisis menunjukkan bahwa area ini memiliki struktur kompresi yang tidak seragam. Dalam dokumen resmi, stempel biasanya tercetak bersama-sama dengan tinta dokumen lainnya, namun dalam kasus ini tampak seperti ditempelkan atau dimodifikasi secara digital.



Tanda tangan yang berada di bagian bawah kiri dan kanan dokumen juga menunjukkan keanehan. Tingkat error yang tinggi di sekitar coretan tanda tangan ini sangat menonjol dibandingkan dengan area sekitarnya. Biasanya, tanda tangan yang ditulis langsung di atas dokumen akan memiliki tingkat error yang konsisten dengan tekstur kertas, tetapi dalam hasil ini tanda tangan terlihat seolah-olah berada di atas lapisan berbeda.

Teks yang berkaitan dengan identitas, seperti tempat lahir “Surakarta” dan jurusan “KEHUTANAN”, juga menunjukkan variasi error yang tidak seragam. Beberapa huruf bahkan terlihat lebih mencolok dalam ELA, seolah-olah dicetak atau ditempelkan secara terpisah dari keseluruhan teks. Hal ini bisa menunjukkan bahwa sebagian informasi identitas pada dokumen ini mengalami perubahan digital.

Tanggal dan lokasi penerbitan dokumen, yang tertera sebagai “YOGYAKARTA, 5 - 11 - 1985”, juga memiliki highlight yang tidak seimbang. Area ini seharusnya tampil seragam dengan sisa dokumen, namun dalam hasil ELA terlihat adanya error level tinggi. Ketidaksesuaian ini membuka kemungkinan tanggal tersebut telah diganti atau diperbaiki secara digital setelah pencetakan.

Selain elemen-elemen utama, tampak bahwa latar belakang dokumen, yakni tekstur kertas, secara umum relatif bebas dari highlight merah mencolok. Ini berarti bahwa bagian latar belakang tampaknya lebih konsisten, dan kemungkinan besar merupakan bagian asli dari dokumen. Ketidakhadiran highlight pada area ini menjadi pembanding penting untuk menunjukkan bahwa bagian-bagian lain memang mengalami perbedaan signifikan dalam struktur digitalnya.

Pada bagian kanan bawah dokumen, terdapat tanda tangan pejabat fakultas dan juga stempel tambahan. Area ini juga menunjukkan ketidaksesuaian dalam error level yang memperkuat dugaan bahwa tanda tangan dan stempel tidak sepenuhnya berasal dari satu proses pencetakan atau pemindaian yang sama dengan dokumen aslinya. Perbedaan ini memperkuat asumsi adanya penggabungan elemen visual secara digital.

Salah satu indikator penting dalam analisis ELA adalah distribusi noise dan highlight. Dalam dokumen ini, distribusi highlight sangat terfokus pada elemen-elemen penting seperti teks, cap, dan foto, sementara bagian latar atau margin dokumen relatif bersih. Distribusi yang tidak merata ini merupakan pola klasik yang sering ditemukan pada dokumen hasil manipulasi digital.

Penting juga memperhatikan bahwa ELA bekerja paling baik jika gambar yang dianalisis merupakan hasil kompresi tunggal dan belum dimodifikasi formatnya secara berulang. Jika gambar yang dianalisis telah mengalami konversi format atau pengeditan berulang, maka hasil ELA bisa menjadi bias. Namun, dalam gambar ini tampaknya masih cukup banyak sinyal visual yang bisa dianalisis.

Distribusi highlight pada bagian kiri bawah dokumen juga menyoroti adanya degradasi atau gangguan di tepi kertas. Meski bisa jadi ini hanya hasil dari kerusakan fisik dokumen asli, jika ditinjau dari ELA, degradasi tersebut menunjukkan ketidakwajaran karena disertai perbedaan error yang tidak konsisten.

Konsistensi antara semua elemen visual adalah kunci dalam menentukan keaslian dokumen. Dalam hal ini, hasil ELA menunjukkan inkonsistensi besar antara teks, foto, stempel, dan tanda tangan. Ketidakkonsistenan ini mencerminkan kemungkinan adanya manipulasi atau penggabungan dari berbagai elemen digital.

ELA juga mengungkap bahwa beberapa area dalam dokumen memiliki highlight berlapis, yakni munculnya kotak merah dalam pola berulang. Ini merupakan indikator umum bahwa bagian tersebut telah disisipkan lebih dari sekali atau mengalami proses editing berlapis.

Dalam konteks dokumen resmi, elemen seperti stempel dan tanda tangan harusnya menjadi satu kesatuan fisik dengan kertas. Namun, dalam gambar ini, kedua elemen tersebut terlihat seolah ditumpuk di atas gambar kertas, seperti yang sering ditemukan pada hasil rekayasa digital atau penyisipan grafis.

Tidak semua highlight dalam ELA menandakan pemalsuan. Namun, ketika highlight terpusat pada elemen krusial dan menghindari area tidak penting, seperti margin atau latar belakang polos, maka kecurigaan menjadi semakin kuat. Distribusi seperti ini jarang muncul dalam dokumen asli.

Hal yang paling menguatkan dugaan adanya rekayasa adalah perbedaan kontras antara elemen-elemen dokumen dan struktur tekstur kertas. Teks asli biasanya menyatu dengan tekstur, tetapi di sini terlihat adanya perbedaan lapisan visual yang cukup jelas melalui ELA.

Secara keseluruhan, hasil ELA menunjukkan indikasi kuat bahwa dokumen ini tidak sepenuhnya autentik secara digital. Banyak elemen penting menunjukkan struktur error yang tidak sesuai dengan latar belakang atau keseluruhan gambar, menimbulkan kecurigaan bahwa dokumen ini telah dimodifikasi atau disusun ulang menggunakan elemen digital yang ditempel secara terpisah.

IMPLEMENTASI ELA DENGAN ADAPTIVE BRIGHTNESS SCALING

Program ini menggunakan metode Error Level Analysis (ELA) sebagai pendekatan utama untuk mendeteksi manipulasi digital pada gambar JPEG. Metode ini memanfaatkan sifat alami dari kompresi lossy JPEG, yaitu bahwa setiap kali gambar disimpan ulang, kualitas gambar akan sedikit menurun. ELA memanfaatkan perbedaan antara gambar asli dan versi yang disimpan ulang untuk menemukan bagian-bagian gambar yang mungkin dimodifikasi, karena bagian yang diedit biasanya memiliki tingkat kompresi yang berbeda dibanding bagian lain.

Proses ELA dimulai dengan menyimpan ulang gambar asli menggunakan kompresi JPEG dengan kualitas tertentu (biasanya 90%). Hal ini dilakukan dalam fungsi `perform_ela`, di mana gambar yang telah dikonversi ke RGB akan disimpan ulang sebagai JPEG pada kualitas 90. File ini bersifat sementara dan digunakan hanya untuk membandingkan perbedaan dengan gambar asli. Gambar hasil simpan ulang ini di-load kembali agar bisa dianalisis secara numerik.

Langkah selanjutnya adalah menghitung perbedaan piksel antara gambar asli dan gambar hasil simpan ulang. Ini dilakukan menggunakan fungsi `ImageChops.difference` dari PIL (Python Imaging Library), yang menghasilkan gambar baru yang mewakili perbedaan intensitas warna dari masing-masing piksel. Area dengan sedikit atau tanpa perbedaan menunjukkan bahwa bagian tersebut tidak berubah, sedangkan area dengan perbedaan besar mengindikasikan adanya kompresi berbeda — dan kemungkinan modifikasi.

Setelah perbedaan dihitung, hasilnya biasanya terlalu gelap atau terlalu lemah untuk dilihat secara kasat mata. Oleh karena itu, dilakukan peningkatan kecerahan (*brightness enhancement*) terhadap gambar perbedaan tersebut. Namun, alih-alih hanya memperkuat menggunakan nilai maksimum dari `diff.getextrema()` seperti pendekatan lama, versi ini menggunakan *adaptive scaling* berdasarkan rata-rata *brightness* dari array piksel hasil perbedaan. Dengan menggunakan *mean brightness*, program dapat secara otomatis menentukan seberapa besar kekuatan

yang diperlukan tanpa menghasilkan over-enhancement yang merusak visual.

Gambar hasil ELA ini kemudian diubah menjadi array NumPy agar dapat diproses secara matematis untuk mendeteksi area mencurigakan. Dengan mengambil nilai maksimum dari masing-masing piksel RGB, didapatkan representasi grayscale dari intensitas kesalahan. Threshold diterapkan (misalnya 40), dan piksel yang nilainya lebih tinggi dari threshold dianggap anomali. Teknik ini membantu menyoroti area yang memiliki tingkat kesalahan tinggi karena adanya kompresi yang tidak seragam.

Untuk membantu visualisasi area yang diduga telah dimanipulasi, dilakukan pembagian citra menjadi blok-blok kecil (patches) berukuran 10x10 piksel. Tiap patch diuji rata-ratanya, dan jika patch tersebut memiliki proporsi piksel mencurigakan lebih dari 40%, maka area tersebut dianggap sebagai kandidat pemalsuan. Area ini kemudian ditandai dengan kotak merah pada gambar asli dan gambar ELA untuk membantu pengguna mengidentifikasi potensi forgeries.

Salah satu aspek penting dari metode ini adalah bahwa ELA tidak membutuhkan data pembandingan eksternal. Berbeda dengan teknik lain seperti deteksi resampling atau sensor pattern noise, ELA hanya bergantung pada file gambar itu sendiri. Oleh karena itu, metode ini berguna untuk pemeriksaan cepat terhadap gambar digital tanpa akses ke kamera asli atau file RAW. Namun, ELA juga memiliki keterbatasan karena bergantung pada format JPEG dan sifat kompresinya.

Untuk memperkaya analisis, program ini juga menampilkan EXIF metadata dari gambar. Metadata ini mencakup informasi seperti jenis kamera, waktu pengambilan, software editing yang digunakan, dan sebagainya. Ketika metadata tidak ditemukan, bisa menjadi indikasi bahwa gambar tersebut telah diproses ulang atau diedit dengan perangkat lunak yang menghapus metadata secara otomatis — sesuatu yang sering dilakukan dalam pemalsuan gambar.

Antarmuka pengguna (GUI) dalam program ini dirancang agar mudah digunakan bahkan oleh pengguna non-teknis. Dengan adanya fitur zoom in dan zoom out, pengguna dapat melihat area yang ditandai secara lebih rinci. Seluruh antarmuka dibuat dalam gaya tabbed interface, memisahkan antara hasil ELA dan metadata, sehingga pengguna bisa berfokus pada visualisasi atau data teknis sesuai kebutuhan.

Dengan demikian, metode ELA dalam program ini berfokus pada prinsip kompresi JPEG, perbedaan tingkat kesalahan kompresi, dan teknik segmentasi blok untuk deteksi anomali. Ditambah dengan adaptive brightness scaling dan visualisasi otomatis, pendekatan ini menjadi lebih modern dan informatif dibanding metode ELA tradisional yang hanya memperlihatkan hasil perbedaan tanpa analisis atau penandaan lebih lanjut.

```
import tkinter as tk
from tkinter import filedialog, ttk
from PIL import Image, ImageTk, ImageChops, ImageEnhance, ImageDraw
import numpy as np
from PIL.ExifTags import TAGS
import os

class ELA_GUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Enhanced ELA Image Forensics")
        self.root.geometry("1200x720")
        self.root.configure(bg="white")

        self.tab_control = ttk.Notebook(self.root)
        self.ela_tab = tk.Frame(self.tab_control, bg="white")
        self.meta_tab = tk.Frame(self.tab_control, bg="white")
        self.tab_control.add(self.ela_tab, text='ELA Analysis')
        self.tab_control.add(self.meta_tab, text='Metadata')
        self.tab_control.pack(expand=1, fill="both")

        self.create_ela_tab()
        self.create_meta_tab()

    def create_ela_tab(self):
        self.original_path = None
        self.ela_image = None
        self.highlighted_image = None
        self.highlighted_ela_image = None
        self.zoom_level = 1.0
        self.zoom_step = 0.2
```

```

        tk.Button(self.ela_tab, text="Upload Image",
command=self.load_image, bg="#4CAF50", fg="white").pack(pady=10)

        self.zoom_frame = tk.Frame(self.ela_tab, bg="#f8f8f8")
        self.zoom_frame.pack(pady=5)

        tk.Button(self.zoom_frame, text="+ Zoom In",
command=self.zoom_in, bg="#607D8B", fg="white").grid(row=0, column=0,
padx=5)

        tk.Button(self.zoom_frame, text="- Zoom Out",
command=self.zoom_out, bg="#607D8B", fg="white").grid(row=0, column=1,
padx=5)

        tk.Button(self.zoom_frame, text="🔍 Highlight Forgery",
command=self.highlight_forgery, bg="#9C27B0", fg="white").grid(row=0,
column=2, padx=5)

        self.canvas = tk.Canvas(self.ela_tab, bg="white")
        self.scroll_x = tk.Scrollbar(self.ela_tab, orient=tk.HORIZONTAL,
command=self.canvas.xview)
        self.scroll_y = tk.Scrollbar(self.ela_tab, orient=tk.VERTICAL,
command=self.canvas.yview)
        self.canvas.configure(xscrollcommand=self.scroll_x.set,
yscrollcommand=self.scroll_y.set)
        self.scroll_x.pack(side=tk.BOTTOM, fill=tk.X)
        self.scroll_y.pack(side=tk.RIGHT, fill=tk.Y)
        self.canvas.pack(expand=True, fill=tk.BOTH)

        self.image_frame = tk.Frame(self.canvas, bg="white")
        self.canvas.create_window((0, 0), window=self.image_frame,
anchor="nw")

        self.original_panel = tk.Label(self.image_frame, text="Original
Image", bg="white")
        self.original_panel.grid(row=0, column=0, padx=20)

        self.overlay_panel = tk.Label(self.image_frame, text="Forgery
Highlight", bg="white")
        self.overlay_panel.grid(row=0, column=1, padx=20)

        self.ela_panel = tk.Label(self.image_frame, text="ELA Image",
bg="white")
        self.ela_panel.grid(row=0, column=2, padx=20)

    def create_meta_tab(self):
        self.meta_text = tk.Text(self.meta_tab, wrap=tk.WORD,
font=("Courier", 10), bg="white", fg="black")
        self.meta_text.pack(expand=1, fill="both", padx=10, pady=10)

    def load_image(self):
        file_path = filedialog.askopenfilename(filetypes=[("JPEG Images",
"*.jpg;*.jpeg")])
        if not file_path:
            return

        self.original_path = file_path
        image = Image.open(file_path).convert("RGB")

```

```

self.ela_image = self.perform_ela(image)

self.highlighted_image = None
self.highlighted_ela_image = None

self.update_zoomed_images()
self.display_metadata(file_path)

def perform_ela(self, image):
    ela_temp = "temp_ela.jpg"
    image.save(ela_temp, 'JPEG', quality=90)
    resaved = Image.open(ela_temp).convert("RGB")

    diff = ImageChops.difference(image, resaved)

    # Use adaptive scaling based on mean brightness
    diff_np = np.array(diff)
    brightness = np.mean(diff_np)
    scale = 255.0 / (brightness + 1e-5) if brightness > 0 else 1

    diff = ImageEnhance.Brightness(diff).enhance(scale)
    os.remove(ela_temp)
    return diff

def highlight_forgery(self, threshold=40):
    if self.ela_image is None:
        return

    ela_np = np.array(self.ela_image)
    mask = np.max(ela_np, axis=2) > threshold

    original = Image.open(self.original_path).convert("RGB")
    draw_original = ImageDraw.Draw(original)

    ela_highlight = self.ela_image.copy()
    draw_ela = ImageDraw.Draw(ela_highlight)

    h, w = mask.shape
    for y in range(0, h - 10, 10):
        for x in range(0, w - 10, 10):
            patch = mask[y:y+10, x:x+10]
            if patch.mean() > 0.4:
                draw_original.rectangle([x, y, x+10, y+10],
outline="red", width=3)
                draw_ela.rectangle([x, y, x+10, y+10], outline="red",
width=3)

    self.highlighted_image = original
    self.highlighted_ela_image = ela_highlight
    self.update_zoomed_images()

def update_zoomed_images(self):
    if not self.original_path:
        return

    original = Image.open(self.original_path).convert("RGB")
    w, h = original.size

```

```

        resized_original = original.resize((int(w * self.zoom_level),
int(h * self.zoom_level)))
        original_tk = ImageTk.PhotoImage(resized_original)
        self.original_panel.configure(image=original_tk)
        self.original_panel.image = original_tk

        if self.highlighted_image:
            resized_overlay = self.highlighted_image.resize((int(w *
self.zoom_level), int(h * self.zoom_level)))
            overlay_tk = ImageTk.PhotoImage(resized_overlay)
            self.overlay_panel.configure(image=overlay_tk)
            self.overlay_panel.image = overlay_tk
        else:
            self.overlay_panel.configure(image='', text="No Highlight
Yet")

        if self.highlighted_ela_image:
            resized_ela = self.highlighted_ela_image.resize((int(w *
self.zoom_level), int(h * self.zoom_level)))
        else:
            resized_ela = self.ela_image.resize((int(w * self.zoom_level),
int(h * self.zoom_level)))

        ela_tk = ImageTk.PhotoImage(resized_ela)
        self.ela_panel.configure(image=ela_tk)
        self.ela_panel.image = ela_tk

        self.image_frame.update_idletasks()
        self.canvas.config(scrollregion=self.canvas.bbox("all"))

    def zoom_in(self):
        self.zoom_level += self.zoom_step
        self.update_zoomed_images()

    def zoom_out(self):
        if self.zoom_level > self.zoom_step:
            self.zoom_level -= self.zoom_step
            self.update_zoomed_images()

    def display_metadata(self, path):
        image = Image.open(path)
        exifdata = image._getexif()
        self.meta_text.delete("1.0", tk.END)

        if not exifdata:
            self.meta_text.insert(tk.END, "No EXIF metadata found.")
            return

        for tag_id, value in exifdata.items():
            tag = TAGS.get(tag_id, tag_id)
            self.meta_text.insert(tk.END, f"{tag:25}: {value}\n")

if __name__ == '__main__':
    root = tk.Tk()
    app = ELA_GUI(root)
    root.mainloop()

```

HASIL IMPLEMENTASI ELA DENGAN ADAPTIVE BRIGHTNESS SCALING ATAS IJAZAH JOKOWI

Hasil Error Level Analysis (ELA) dengan adaptive brightness scaling pada gambar ijazah Jokowi ini memberikan tampilan yang jauh lebih jelas terhadap area-area yang mengalami perbedaan signifikan dalam tingkat kompresi JPEG. Teknik ini meningkatkan kontras antara bagian yang termodifikasi dan bagian yang tidak, sehingga kita bisa mengidentifikasi indikasi potensi manipulasi digital dengan lebih teliti. Pada tampilan yang ditunjukkan, kita bisa melihat sebaran kotak merah sangat padat dan merata di area teks, lambang, foto, dan stempel, namun tidak menyeluruh di latar belakang kertas.

Pertama-tama, area yang paling mencolok adalah bagian tengah dokumen yang menampilkan lambang Universitas Gadjah Mada. Lambang tersebut menunjukkan kepadatan error yang tinggi, bahkan dalam format adaptive brightness, distribusinya tetap konsisten sangat terang dan padat, mengindikasikan kemungkinan bahwa elemen ini dimasukkan secara terpisah atau mengalami modifikasi digital, bukan bagian asli dari dokumen yang dipindai.

Teks besar “IJAZAH” juga memiliki konsentrasi error sangat tinggi, dengan warna merah mencolok dan blok-blok yang sangat terstruktur. Hal ini menunjukkan bahwa teks ini mungkin tidak seotentik latar belakangnya. Dalam dokumen asli yang dicetak sekaligus, teks seperti ini seharusnya memiliki tingkat kompresi seragam dengan sekitarnya. Ketimpangan ini menjadi salah satu indikasi adanya pemrosesan ulang atau overlay teks.

Seluruh bagian teks lainnya seperti “UNIVERSITAS GADJAH MADA”, “FAKULTAS KEHUTANAN”, dan “TELAH LULUS” juga memperlihatkan distribusi error yang tinggi. Ini menunjukkan bahwa kemungkinan besar teks ini ditambahkan atau diedit ulang secara digital. Tingkat highlight yang tinggi pada huruf demi huruf menyiratkan pemisahan struktur dari latar belakang dokumen.

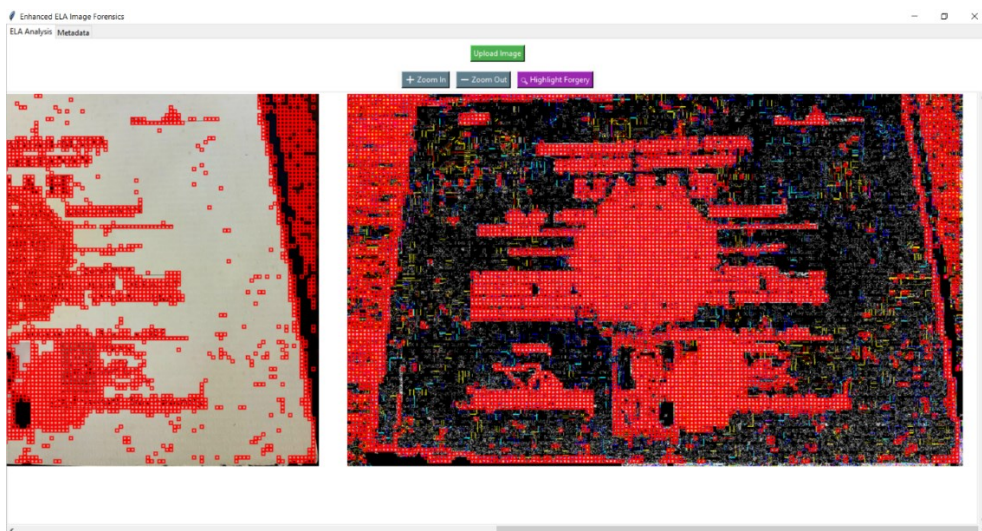
Analisis semakin menarik ketika melihat teks identitas seperti “Sarjana Kehutanan” dan tempat lahir “Di Surakarta”. Semua bagian ini juga memperlihatkan hasil highlight yang padat dan konsisten dalam warna merah terang. Ini menunjukkan bahwa bagian ini mungkin tidak berasal dari pencetakan awal yang sama, melainkan disisipkan setelah pemindaian atau pengeditan digital berlangsung.



Tanggal dan lokasi penerbitan dokumen, yaitu “YOGYAKARTA, 5 – 11 – 1985”, juga menunjukkan tingkat error yang tidak wajar. Dalam dokumen sah yang utuh, tanggal dan lokasi seperti ini umumnya memiliki tingkat kompresi serupa dengan teks lainnya. Namun di sini, ketimpangan intensitas error menunjukkan bahwa bagian ini mengalami modifikasi atau setidaknya disisipkan ulang.

Bagian bawah dokumen memperlihatkan foto pemilik ijazah beserta stempel dan tanda tangan. Foto ini menunjukkan intensitas error yang sangat tinggi, bahkan lebih tinggi dari teks di sekitarnya. Hal ini menunjukkan bahwa kemungkinan besar foto ini tidak tertanam secara fisik di dokumen asli, melainkan ditempelkan atau ditambahkan setelah proses digitalisasi.

Stempel merah bulat yang biasanya menunjukkan legalitas dokumen juga memperlihatkan highlight sangat pekat. Dalam hasil ELA ini, distribusinya terlalu terang dan konsisten untuk dianggap bagian alami dari dokumen yang dicetak dan kemudian dipindai. Ini mengindikasikan bahwa stempel tersebut bisa jadi hasil overlay atau ditempel secara digital.



Tanda tangan di bawah foto dan nama pejabat (Dekan) tampak mengalami tingkat error yang tinggi. Highlight ini menunjukkan bahwa tanda tangan kemungkinan ditambahkan secara terpisah dari dokumen utama. Normalnya, tinta tanda tangan fisik akan menghasilkan pola error yang lebih tersebar dan tidak seterstruktur dalam ELA, karena interaksi langsung antara tinta dan kertas.

Cap bea materai di sebelah foto, yang berupa stempel hijau 100 rupiah, juga menunjukkan hasil yang tidak natural. Kompresinya terlalu terlokalisasi dan padat dalam highlight, menandakan bahwa objek ini kemungkinan dimasukkan atau diproses ulang secara digital. Dalam dokumen autentik, cap biasanya akan menyatu dengan tekstur kertas.

Perlu dicermati pula bagian pinggir dokumen yang dalam adaptive brightness analysis tampak lebih gelap dan kurang highlight dibanding

bagian tengah. Ini justru menunjukkan bahwa area pinggir lebih otentik dan belum mengalami modifikasi digital berarti. Area yang tidak disentuh editor biasanya tampil lebih seragam dalam hasil ELA.

Distribusi highlight secara horizontal menunjukkan pola yang tidak merata, yang menyiratkan bahwa dokumen ini mungkin telah melalui proses layering. Penggunaan berbagai lapisan digital untuk menyusun teks, gambar, dan tanda tangan secara terpisah bisa menghasilkan struktur seperti ini saat dianalisis dengan ELA berbasis adaptive brightness.

Warna merah yang sangat menyala dan blok-blok kecil yang rapat pada teks menunjukkan bahwa resolusi kompresi JPEG antara objek dan latar belakang tidak sinkron. Ini biasanya terjadi saat seseorang mengedit teks pada dokumen menggunakan software pengolah gambar kemudian menyimpan ulang dalam format lossy seperti JPEG.

Sebagian area juga menunjukkan highlight blok yang melampaui batas huruf, terutama pada bagian tengah dokumen. Ini dapat mengindikasikan adanya penempelan layer transparan atau penggabungan objek dengan metode digital tertentu, bukan hasil dari pemindaian satu dokumen yang utuh.

Tanda-tanda ini diperkuat oleh tampilan warna hitam dan beragam warna digital seperti biru, kuning, dan hijau dalam mode paling akhir ELA. Warna-warni ini biasanya tidak muncul pada dokumen asli yang hanya mengalami kompresi biasa, dan lebih sering diasosiasikan dengan penyisipan elemen digital atau proses pascaproduksi grafis.

Analisis ini juga menunjukkan bahwa perbedaan error antara teks dan latar belakang jauh lebih kontras dibanding dokumen biasa. Artinya, latar belakang tampak lebih natural, sementara teks dan objek lain menonjol dalam error level. Ketimpangan seperti ini menjadi dasar utama dalam mencurigai adanya manipulasi.

Pada bagian kanan dokumen, terutama dekat dengan tanda tangan dan nama “Prof. Dr. Soenardi”, perbedaan error sangat tajam dan bahkan beberapa pixel memperlihatkan noise pixelated. Ini indikasi klasik bahwa tanda tangan itu mungkin ditambahkan menggunakan brush tool atau teknik penyisipan bitmap dalam pengeditan grafis.

Ciri lain yang menonjol adalah kurangnya noise di bagian permukaan kertas yang polos, namun tingginya noise di sekitar objek visual utama. Dalam adaptive brightness, kertas polos seharusnya memperlihatkan sedikit noise, dan ini sesuai. Tetapi saat objek di atasnya terlalu terang, kemungkinan besar objek tersebut berasal dari proses berbeda dibanding kertas.

Beberapa bagian teks terlihat sedikit buram namun tetap menunjukkan error tinggi. Ini sering terjadi jika teks ditambahkan secara digital, kemudian diterapkan filter agar menyerupai teks asli dalam dokumen scan. Namun filter ini tidak menyembunyikan jejak error yang diungkap ELA.

Secara keseluruhan, ELA dengan adaptive brightness scaling memberikan bukti visual yang cukup kuat untuk menyimpulkan bahwa dokumen ijazah ini kemungkinan besar telah mengalami beberapa bentuk manipulasi digital. Elemen-elemen penting seperti teks utama, foto, stempel, dan tanda tangan menunjukkan pola error yang tidak alami dan berbeda dari latar belakang dokumen.

IMPLEMENTASI ELA DENGAN CLAHE (CONTRAST LIMITED ADAPTIVE HISTOGRAM EQUALIZATION)

Program di atas menggunakan metode Enhanced Error Level Analysis (ELA) dalam GUI berbasis Tkinter untuk mendeteksi tanda-tanda manipulasi digital pada gambar JPEG. Secara spesifik, fungsi `perform_ela` mengimplementasikan metode visualisasi perbedaan antara gambar asli dan hasil kompresi ulang menggunakan teknik yang lebih akurat dan halus dibandingkan metode ELA konvensional. Proses ini sangat berguna

dalam digital forensics untuk mengungkap apakah suatu gambar telah diedit secara tidak sah.

Pertama, gambar dibuka menggunakan PIL dan dikonversi ke format OpenCV BGR untuk pengolahan citra yang lebih canggih. Proses kompresi ulang dilakukan dengan kualitas JPEG 90 menggunakan OpenCV, kemudian gambar asli dan yang telah dikompresi dibandingkan menggunakan cv2.absdiff. Fungsi ini memberikan representasi visual terhadap setiap piksel yang berubah akibat kompresi, yang sangat sensitif terhadap manipulasi lokal seperti tempelan digital (cloning) atau pengubahan isi gambar.

Setelah mendapatkan citra perbedaan tersebut, citra diubah ke format grayscale untuk menekankan hanya informasi intensitas, bukan warna. Kemudian diterapkan CLAHE (Contrast Limited Adaptive Histogram Equalization), sebuah metode yang meningkatkan kontras lokal pada citra. Ini sangat penting karena manipulasi digital sering menghasilkan blok atau area yang secara kontras sangat berbeda dari sekitarnya, tetapi mungkin tidak terlihat jelas secara visual tanpa peningkatan seperti ini.

Langkah berikutnya adalah Gaussian blur, yang digunakan untuk mengurangi noise yang mungkin muncul akibat kompresi dan perhitungan perbedaan. Blur ini membuat pola manipulasi menjadi lebih mudah dilihat dengan menghaluskan gangguan kecil namun tetap mempertahankan perbedaan yang signifikan dalam citra. Hal ini membuat hasil analisis lebih stabil dan bersih saat ditampilkan.

Setelah gambar diperhalus, dilakukan normalisasi agar nilai intensitas piksel disesuaikan dalam rentang 0-255 secara penuh. Hal ini membuat perbedaan kecil menjadi lebih terlihat oleh mata manusia. Kemudian hasil akhirnya dikonversi kembali ke format PIL untuk ditampilkan dalam antarmuka GUI Tkinter. Konversi ini penting karena Tkinter membutuhkan format PIL untuk menampilkan gambar dalam widget Label.

Fungsi `highlight_forgery` memanfaatkan hasil ELA untuk membuat visualisasi blok-blok yang menunjukkan potensi manipulasi. Ia memeriksa setiap patch 10x10 piksel dalam hasil ELA dan mengevaluasi apakah rata-rata nilai intensitasnya melebihi ambang tertentu ($\text{threshold}=40$). Jika ya, maka area tersebut dianggap mencurigakan dan diberi kotak merah baik di atas gambar asli maupun hasil ELA, memperjelas lokasi manipulasi kepada pengguna.

Penggunaan patch kecil dan metode thresholding ini membuat sistem mampu mendeteksi manipulasi pada level sangat lokal, seperti perubahan pada teks, stempel, tanda tangan, atau wajah dalam dokumen digital. Ini sangat bermanfaat untuk forensik dokumen seperti ijazah, paspor, atau dokumen identitas yang mungkin dimanipulasi untuk tujuan ilegal.

Selain itu, GUI ini memberikan fungsi zoom dan navigasi yang nyaman, memudahkan pengguna untuk meneliti bagian gambar secara detail. Ini sangat penting dalam analisis forensik, di mana identifikasi bagian kecil dari manipulasi bisa menjadi bukti penting dalam investigasi hukum atau penelitian keaslian.

Tak kalah penting, tab metadata (`create_meta_tab` dan `display_metadata`) memberikan akses terhadap informasi EXIF, yaitu data tambahan yang tertanam dalam file JPEG seperti model kamera, waktu pengambilan gambar, lokasi GPS, dan lainnya. Data ini sangat penting dalam proses otentikasi karena dapat dibandingkan dengan metadata dari sumber asli.

Secara keseluruhan, metode ini memadukan teknik rekonstruksi ELA berbasis OpenCV, peningkatan kontras lokal adaptif, perataan noise dengan Gaussian blur, serta highlighting cerdas berbasis mask dan thresholding, dalam satu sistem GUI yang intuitif. Ini membuatnya sangat cocok digunakan dalam proses digital forensics modern, khususnya dalam mendeteksi manipulasi citra yang sulit dilihat dengan mata telanjang.

```

import tkinter as tk
from tkinter import filedialog, ttk
from PIL import Image, ImageTk, ImageDraw
import numpy as np
from PIL.ExifTags import TAGS
import os
import cv2 # Pastikan OpenCV sudah terinstal

class ELA_GUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Enhanced ELA Image Forensics")
        self.root.geometry("1200x720")
        self.root.configure(bg="white")

        self.tab_control = ttk.Notebook(self.root)
        self.ela_tab = tk.Frame(self.tab_control, bg="white")
        self.meta_tab = tk.Frame(self.tab_control, bg="white")
        self.tab_control.add(self.ela_tab, text='ELA Analysis')
        self.tab_control.add(self.meta_tab, text='Metadata')
        self.tab_control.pack(expand=1, fill="both")

        self.create_ela_tab()
        self.create_meta_tab()

    def create_ela_tab(self):
        self.original_path = None
        self.ela_image = None
        self.highlighted_image = None
        self.highlighted_ela_image = None
        self.zoom_level = 1.0
        self.zoom_step = 0.2

        tk.Button(self.ela_tab, text="Upload Image",
command=self.load_image, bg="#4CAF50", fg="white").pack(pady=10)

        self.zoom_frame = tk.Frame(self.ela_tab, bg="#f8f8f8")
        self.zoom_frame.pack(pady=5)

        tk.Button(self.zoom_frame, text="+ Zoom In",
command=self.zoom_in, bg="#607D8B", fg="white").grid(row=0, column=0,
padx=5)

        tk.Button(self.zoom_frame, text="— Zoom Out",
command=self.zoom_out, bg="#607D8B", fg="white").grid(row=0, column=1,
padx=5)

        tk.Button(self.zoom_frame, text="🔍 Highlight Forgery",
command=self.highlight_forgery, bg="#9C27B0", fg="white").grid(row=0,
column=2, padx=5)

        self.canvas = tk.Canvas(self.ela_tab, bg="white")
        self.scroll_x = tk.Scrollbar(self.ela_tab, orient=tk.HORIZONTAL,
command=self.canvas.xview)
        self.scroll_y = tk.Scrollbar(self.ela_tab, orient=tk.VERTICAL,
command=self.canvas.yview)
        self.canvas.configure(xscrollcommand=self.scroll_x.set,
yscrollcommand=self.scroll_y.set)

```

```

self.scroll_x.pack(side=tk.BOTTOM, fill=tk.X)
self.scroll_y.pack(side=tk.RIGHT, fill=tk.Y)
self.canvas.pack(expand=True, fill=tk.BOTH)

self.image_frame = tk.Frame(self.canvas, bg="white")
self.canvas.create_window((0, 0), window=self.image_frame,
anchor="nw")

self.original_panel = tk.Label(self.image_frame, text="Original
Image", bg="white")
self.original_panel.grid(row=0, column=0, padx=20)

self.overlay_panel = tk.Label(self.image_frame, text="Forgery
Highlight", bg="white")
self.overlay_panel.grid(row=0, column=1, padx=20)

self.ela_panel = tk.Label(self.image_frame, text="ELA Image",
bg="white")
self.ela_panel.grid(row=0, column=2, padx=20)

def create_meta_tab(self):
    self.meta_text = tk.Text(self.meta_tab, wrap=tk.WORD,
font=("Courier", 10), bg="white", fg="black")
    self.meta_text.pack(expand=1, fill="both", padx=10, pady=10)

def load_image(self):
    file_path = filedialog.askopenfilename(filetypes=[("JPEG Images",
"*.jpg;*.jpeg")])
    if not file_path:
        return

    self.original_path = file_path
    image = Image.open(file_path).convert("RGB")
    self.ela_image = self.perform_ela(image)

    self.highlighted_image = None
    self.highlighted_ela_image = None

    self.update_zoomed_images()
    self.display_metadata(file_path)

def perform_ela(self, pil_image):
    # Convert to OpenCV
    np_image = np.array(pil_image)
    bgr = cv2.cvtColor(np_image, cv2.COLOR_RGB2BGR)
    temp_path = "temp_ela.jpg"
    cv2.imwrite(temp_path, bgr, [int(cv2.IMWRITE_JPEG_QUALITY), 90])
    compressed = cv2.imread(temp_path)

    # ELA using OpenCV
    ela = cv2.absdiff(bgr, compressed)
    gray = cv2.cvtColor(ela, cv2.COLOR_BGR2GRAY)

    # CLAHE
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    enhanced = clahe.apply(gray)

```

```

# Gaussian Blur
blurred = cv2.GaussianBlur(enhanced, (5, 5), 0)

# Normalize
norm = cv2.normalize(blurred, None, 0, 255, cv2.NORM_MINMAX)

# Convert back to PIL
final = Image.fromarray(norm).convert("RGB")
os.remove(temp_path)
return final

def highlight_forgery(self, threshold=40):
    if self.ela_image is None:
        return

    ela_np = np.array(self.ela_image)
    mask = np.max(ela_np, axis=2) > threshold

    original = Image.open(self.original_path).convert("RGB")
    draw_original = ImageDraw.Draw(original)

    ela_highlight = self.ela_image.copy()
    draw_ela = ImageDraw.Draw(ela_highlight)

    h, w = mask.shape
    for y in range(0, h - 10, 10):
        for x in range(0, w - 10, 10):
            patch = mask[y:y+10, x:x+10]
            if patch.mean() > 0.4:
                draw_original.rectangle([x, y, x+10, y+10],
outline="red", width=3)
                draw_ela.rectangle([x, y, x+10, y+10], outline="red",
width=3)

    self.highlighted_image = original
    self.highlighted_ela_image = ela_highlight
    self.update_zoomed_images()

def update_zoomed_images(self):
    if not self.original_path:
        return

    original = Image.open(self.original_path).convert("RGB")
    w, h = original.size

    resized_original = original.resize((int(w * self.zoom_level),
int(h * self.zoom_level)))
    original_tk = ImageTk.PhotoImage(resized_original)
    self.original_panel.configure(image=original_tk)
    self.original_panel.image = original_tk

    if self.highlighted_image:
        resized_overlay = self.highlighted_image.resize((int(w *
self.zoom_level), int(h * self.zoom_level)))
        overlay_tk = ImageTk.PhotoImage(resized_overlay)
        self.overlay_panel.configure(image=overlay_tk)
        self.overlay_panel.image = overlay_tk

```



```

        else:
            self.overlay_panel.configure(image='', text="No Highlight
Yet")

        if self.highlighted_ela_image:
            resized_ela = self.highlighted_ela_image.resize((int(w *
self.zoom_level), int(h * self.zoom_level)))
        else:
            resized_ela = self.ela_image.resize((int(w * self.zoom_level),
int(h * self.zoom_level)))

        ela_tk = ImageTk.PhotoImage(resized_ela)
        self.ela_panel.configure(image=ela_tk)
        self.ela_panel.image = ela_tk

        self.image_frame.update_idletasks()
        self.canvas.config(scrollregion=self.canvas.bbox("all"))

    def zoom_in(self):
        self.zoom_level += self.zoom_step
        self.update_zoomed_images()

    def zoom_out(self):
        if self.zoom_level > self.zoom_step:
            self.zoom_level -= self.zoom_step
            self.update_zoomed_images()

    def display_metadata(self, path):
        image = Image.open(path)
        exifdata = image._getexif()
        self.meta_text.delete("1.0", tk.END)

        if not exifdata:
            self.meta_text.insert(tk.END, "No EXIF metadata found.")
            return

        for tag_id, value in exifdata.items():
            tag = TAGS.get(tag_id, tag_id)
            self.meta_text.insert(tk.END, f"{tag:25}: {value}\n")

if __name__ == '__main__':
    root = tk.Tk()
    app = ELA_GUI(root)
    root.mainloop()

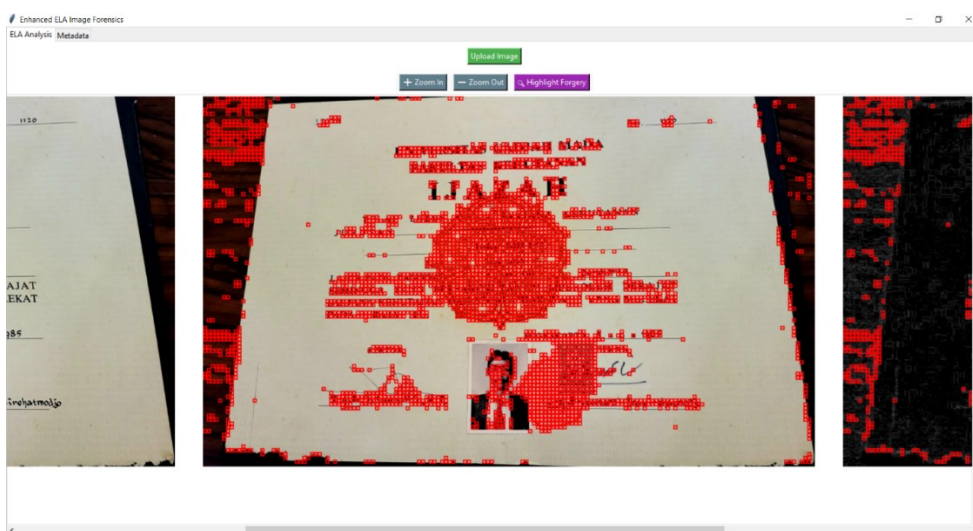
```

HASIL IMPLEMENTASI ELA DENGAN CLAHE (CONTRAST LIMITED ADAPTIVE HISTOGRAM EQUALIZATION) ATAS IJAZAH JOKOWI

Hasil analisis Error Level Analysis (ELA) dengan teknik CLAHE (Contrast Limited Adaptive Histogram Equalization) pada gambar ijazah Jokowi ini memberikan tingkat kontras yang jauh lebih tinggi terhadap perbedaan tingkat kompresi JPEG di seluruh dokumen. CLAHE bekerja dengan cara

memperkuat kontras lokal dalam bagian-bagian kecil dari gambar tanpa menyebabkan efek over-amplifikasi pada noise, sehingga sangat cocok digunakan untuk mendeteksi manipulasi digital yang tersembunyi dalam dokumen berformat kompleks seperti ijazah.

Dalam tampilan hasil pertama, kita langsung melihat adanya penebalan pada area teks "IJAZAH", lambang UGM di tengah, serta blok teks lainnya. Tingkat highlight merah pada area ini sangat tinggi dan seragam, yang mengindikasikan bahwa teks dan lambang tersebut tidak memiliki konsistensi kompresi dengan latar belakang kertas. Ketidaksesuaian ini menjadi indikasi awal adanya penyisipan elemen atau pemrosesan digital terhadap bagian-bagian tersebut.



Bagian tengah dokumen, tempat lambang Universitas Gadjah Mada berada, menunjukkan intensitas error yang sangat tinggi dan padat, dengan persebaran kotak merah yang tertata rapat. Biasanya, jika lambang ini merupakan bagian dari dokumen asli yang dipindai secara utuh, maka tingkat error-nya akan serupa dengan latar belakang. Tetapi hasil ELA dengan CLAHE ini menunjukkan kontras sangat tajam, memperkuat dugaan bahwa lambang tersebut mungkin telah diedit atau ditambahkan secara digital.

Foto pemilik ijazah di bagian bawah tampak sangat mencolok pada hasil ELA ini. CLAHE memperjelas batas antara wajah, pakaian, dan latar belakang foto, serta perbedaan kompresi yang ekstrem dibandingkan bagian kertas ijazah di sekitarnya. Hal ini mengarah pada asumsi bahwa foto tersebut kemungkinan besar ditempelkan setelah dokumen dibuat atau disisipkan secara digital.

Cap meterai 100 rupiah di sebelah kiri bawah foto juga memperlihatkan tingkat error yang tinggi, namun highlight-nya tidak menyatu dengan kertas dasar. CLAHE mengungkap ketidakhomogenan kontras pada area meterai, seolah-olah bagian tersebut berasal dari sumber berbeda. Ini memberikan indikasi bahwa elemen meterai tidak berasal dari pemindaian tunggal dokumen fisik, melainkan mungkin hasil digitalisasi tambahan.



Tanda tangan di bawah teks "REKTOR" juga tampak mencolok dalam hasil ELA ini. CLAHE menyoroti bahwa tanda tangan tersebut memiliki kontras tinggi yang tidak menyatu secara halus dengan permukaan kertas. Biasanya tanda tangan yang ditulis langsung dengan tinta akan menunjukkan transisi error yang alami, tetapi pada kasus ini justru menampilkan tepi yang sangat tajam—kemungkinan disisipkan menggunakan brush digital.

Hal yang sama terlihat pada tanda tangan di sisi kanan bawah (DEKAN). Tanda tangan ini pun tampak terpisah secara tekstural dari latar belakang. CLAHE memperjelas bahwa distribusi error pada tanda tangan tersebut sangat padat dan tidak tersebar, yang merupakan tanda khas dari penyisipan digital, bukan hasil tulis tangan fisik.

Bagian tulisan "YOGYAKARTA, 5 – 11 – 1985" yang menunjukkan lokasi dan tanggal penerbitan ijazah, juga menampilkan anomali. CLAHE mengungkapkan adanya ketidakhomogenan pada huruf dan angka, serta kontur pixelated pada angka tahun "1985". Perbedaan ini mengisyaratkan bahwa tanggal bisa jadi telah dimodifikasi atau ditambahkan ulang pada file digital sebelum dicetak atau dipindai ulang.

Ketidakwajaran juga muncul pada area teks "Di Surakarta", yang tampak mencolok dibandingkan teks sekitarnya. CLAHE menampilkan blok merah terang yang sangat padat di sekitar teks ini, dan ini menyiratkan bahwa teks tersebut memiliki struktur digital berbeda, memperkuat asumsi bahwa lokasi kelahiran mungkin diubah dari informasi asli.

Blok teks lain seperti "TELAH LULUS", "SARJANA KEHUTANAN", serta keterangan panjang tentang kelulusan juga menunjukkan highlight error yang sangat terfokus. Dalam CLAHE, bagian-bagian ini tidak terlihat sebagai bagian homogen dari kertas, tapi sebagai lapisan teks yang memiliki profil kompresi berbeda, yang menunjukkan kemungkinan penyisipan atau rekonstruksi teks secara digital.

Salah satu elemen paling penting dalam forensik dokumen adalah latar belakang kertas. Pada hasil ini, latar belakang tampak relatif bersih dengan noise yang wajar, sementara hampir seluruh bagian teks, lambang, stempel, foto, dan tanda tangan menunjukkan highlight CLAHE yang kontras. Ini berarti latar belakang kemungkinan merupakan bagian asli yang dipindai, sedangkan elemen-elemen lain ditambahkan atau diedit secara terpisah.

Perbedaan sangat jelas antara elemen asli dan yang ditambahkan bisa dilihat dari area sudut dokumen, terutama sisi kiri bawah dan kanan atas.

Dalam CLAHE, bagian ini menunjukkan intensitas error yang sangat rendah, mendekati alami. Kontrasnya dengan teks atau stempel yang memiliki error tinggi menjadi bukti visual bahwa tidak semua bagian gambar memiliki asal-usul kompresi yang sama.

Dalam tampilan terakhir ELA-CLAHE, pola pixel merah yang menutupi sebagian besar teks dan objek penting mengindikasikan ketidakwajaran secara menyeluruh dalam penyusunan dokumen ini. Tidak ada gradien error alami, yang seharusnya muncul jika semua elemen berasal dari proses scan tunggal. Sebaliknya, elemen-elemen visual menunjukkan profil kompresi blok yang berbeda.

CLAHE juga mengungkap noise berbentuk linear pada area batas antara teks dan latar, yang mengindikasikan kemungkinan besar bahwa teks disisipkan dengan bantuan perangkat lunak desain grafis. Perbedaan kontras ekstrem antara huruf-huruf dan area putih kosong memperkuat argumen ini.

Uniknya, bagian bawah dokumen, terutama area nama “Prof. Dr. Soenardi S.” hingga cap stempel merah di dekatnya, memiliki highlight ELA yang sangat padat dan terlokalisir. Ini menunjukkan bahwa kemungkinan bagian tersebut dimasukkan melalui metode penggabungan digital, misalnya overlay bitmap pada gambar scan.

Selain itu, CLAHE memperlihatkan keberadaan artefak edge enhancement di sekitar beberapa teks, yang merupakan tanda bahwa sharpening atau penyuntingan ketajaman mungkin telah dilakukan untuk membuat dokumen tampak lebih tajam dari hasil scan biasa. Ini menjadi indikasi tambahan adanya proses rekonstruksi atau pengeditan.

Dari segi pencahayaan, CLAHE menunjukkan bahwa tidak ada transisi bayangan halus antara area teks dan latar belakang, padahal dokumen fisik asli biasanya menampilkan bayangan mikro akibat ketebalan tinta, emboss, atau tekstur fisik. Ketidakhadiran ini menunjukkan bahwa kemungkinan besar dokumen merupakan hasil kompilasi digital.

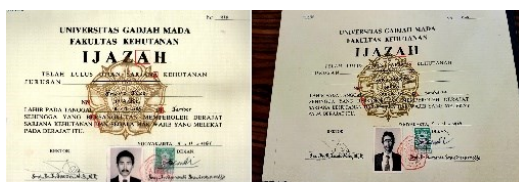
Secara umum, hasil ELA menggunakan CLAHE sangat mendukung dugaan bahwa dokumen ini tidak otentik secara keseluruhan. Ketidaksamaan kontras kompresi antara latar belakang dan semua elemen penting (teks, tanda tangan, stempel, foto) memperkuat argumen bahwa dokumen ini mengalami modifikasi digital atau bahkan rekayasa penuh.

Teknik CLAHE bukan hanya memperjelas distribusi error JPEG, tetapi juga memetakan ketidakkonsistenan kontras mikro yang tidak bisa dilihat dalam metode ELA biasa. Dalam konteks ini, hasil yang ditampilkan menjadi sangat signifikan karena memperlihatkan pola sistematis penyisipan elemen visual, bukan hasil dari satu proses pemindaian utuh.

Kesimpulannya, dokumen ini secara visual tidak menunjukkan keseragaman dalam struktur kompresi berdasarkan hasil ELA dengan CLAHE. Ketidakwajaran distribusi error, perbedaan signifikan antara elemen penting dan latar belakang, serta konsistensi highlight pada bagian-bagian tertentu semuanya mengarah pada kesimpulan kuat bahwa ijazah tersebut besar kemungkinan telah dimanipulasi secara digital.

OVERLAPPING DETECTION

TENTANG OVERLAPPING DETECTION



Overlapping detection atau deteksi tumpang tindih merupakan salah satu metode penting dalam forensik digital, khususnya dalam menganalisis keaslian dan keidentikan dua dokumen visual. Metode ini bertujuan untuk mengevaluasi apakah elemen-elemen tertentu dari dua gambar atau dokumen digital memiliki pola tumpang tindih yang konsisten. Ini sangat berguna ketika dua dokumen harus dibandingkan untuk

mengetahui apakah mereka berasal dari sumber yang sama, atau jika salah satu di antaranya merupakan hasil rekayasa digital.

Dalam konteks forensik ijazah, sertifikat, atau dokumen hukum lainnya, overlapping detection dapat digunakan untuk memverifikasi posisi relatif antara stempel, tanda tangan, dan teks. Misalnya, jika dalam satu dokumen asli stempel menimpa sebagian huruf nama, maka salinan yang identik seharusnya menunjukkan pola tumpang tindih yang sama. Jika tidak, maka ada kemungkinan bahwa salah satu elemen, misalnya

stempel, telah ditambahkan kemudian melalui proses digital seperti pengeditan gambar.

Teknik ini sangat berguna dalam menganalisis pemalsuan digital, terutama ketika pemalsu mencoba menyalin elemen-elemen otentik dari dokumen asli, lalu menempelkannya ke dokumen palsu. Dengan membandingkan posisi dan intensitas tumpang tindih antara dua dokumen, penyidik digital dapat mendeteksi ketidaksesuaian yang tidak kasat mata. Misalnya, sedikit perbedaan dalam jarak antara stempel dan teks atau hilangnya sebagian detail karena proses penempelan bisa menjadi bukti manipulasi.

Deteksi tumpang tindih juga berguna dalam membandingkan dokumen yang dicetak secara massal, seperti ijazah atau surat keputusan resmi. Jika dua dokumen memiliki nomor seri yang berbeda tetapi pola tumpang tindih antara teks dan logo persis sama, maka ada kecurigaan bahwa salah satu dokumen adalah hasil duplikasi yang tidak sah. Dalam hal ini, overlapping detection dapat digunakan untuk membuktikan bahwa satu dokumen adalah hasil fotokopi atau manipulasi dari dokumen lain.

Metode ini biasanya melibatkan segmentasi gambar menjadi beberapa komponen warna atau bentuk, kemudian menghitung luas atau persentase piksel yang saling menutupi. Proses ini dapat dilakukan secara otomatis menggunakan citra digital dengan ruang warna HSV atau metode threshold biner, seperti pada huruf yang berwarna hitam dan stempel yang berwarna coklat atau merah. Nilai persentase overlapping yang terlalu rendah atau terlalu tinggi dari ekspektasi normal bisa menjadi indikator kecurangan.

Dalam penyelidikan forensik dokumen, nilai overlapping juga digunakan untuk membangun ground truth — artinya, digunakan untuk mengkalibrasi dan membandingkan banyak dokumen otentik untuk menentukan standar keaslian. Dari sini, penyidik bisa menentukan toleransi batas wajar tumpang tindih dalam dokumen yang sah. Jika sebuah dokumen menyimpang dari pola tersebut, maka dapat dicurigai sebagai tidak otentik.

Metode overlapping detection juga sangat berguna untuk membuktikan modus operandus dalam pemalsuan. Misalnya, pelaku dapat menghapus teks asli dengan alat digital, lalu menyisipkan teks baru dan menambahkan stempel dari dokumen lain. Dengan overlapping detection, pola yang tak alami—seperti huruf yang tidak tertimpa sama sekali atau stempel yang tidak menutupi bagian teks penting—bisa dibongkar dan digunakan sebagai bukti di pengadilan.

Kelebihan lain dari overlapping detection adalah bahwa metode ini tidak memerlukan basis data teks atau OCR, sehingga bisa digunakan untuk dokumen dalam berbagai bahasa, bahkan tulisan tangan. Sistem hanya bergantung pada pola visual dan distribusi spasial antar objek dalam gambar, membuatnya sangat fleksibel untuk diterapkan dalam berbagai konteks.

Di ranah akademik atau pemerintahan, overlapping detection dapat membantu mendeteksi duplikasi ijazah, sertifikat pelatihan, atau dokumen izin yang dipalsukan. Penyidik bisa dengan mudah mendeteksi apakah suatu stempel atau tanda tangan adalah hasil salinan dari dokumen lain hanya dengan mencocokkan seberapa konsisten elemen-elemen tersebut bertumpang tindih pada beberapa dokumen.

Secara keseluruhan, overlapping detection memberikan kontribusi besar dalam verifikasi visual yang tidak bisa dideteksi secara tekstual. Ia bekerja sebagai lapisan tambahan dalam forensik digital visual, memberikan bukti objektif dan kuantitatif dalam membandingkan dokumen yang secara visual terlihat identik tetapi secara spasial menunjukkan ketidaksesuaian. Dengan begitu, metode ini merupakan alat yang sangat penting dalam proses otentikasi dokumen digital dan analog dalam ranah hukum, pendidikan, dan pemerintahan.

IMPLEMENTASI OVERLAPPING DETECTION PADA RUANG WARNA HSV (*Hue, Saturation, and Value*)

Metode utama yang digunakan dalam kode ini adalah deteksi overlapping atau tumpang tindih antara huruf berwarna hitam dan logo berwarna coklat menggunakan ruang warna HSV. Saat pengguna memilih dan

memotong area pada gambar, sistem mengubah bagian gambar tersebut dari ruang warna BGR menjadi HSV, karena HSV memudahkan dalam membedakan warna berdasarkan rona (hue), saturasi, dan nilai intensitas cahaya. Setelah itu, dibuat dua masking menggunakan batas bawah dan atas nilai HSV: satu untuk mendeteksi huruf hitam (mask_huruf) dan satu lagi untuk mendeteksi logo berwarna coklat (mask_logo). Dengan dua masker ini, sistem melakukan operasi logika bitwise_and untuk menghitung jumlah piksel huruf yang tertimpa oleh logo.

Setelah piksel-piksel tertimpa dihitung, program menghitung persentase overlapping dengan membagi jumlah piksel tumpang tindih terhadap jumlah total piksel huruf, lalu dikalikan 100. Hasil persentase ini ditampilkan di GUI sebagai indikator visual sejauh mana logo menutupi teks. Jika nilai overlapping terlalu kecil atau terlalu besar dari normal, ini bisa menjadi indikasi bahwa logo atau teks telah dimanipulasi. Dengan kata lain, metode ini dapat digunakan dalam forensik digital untuk menilai keaslian dokumen dengan mendeteksi pola yang tidak wajar dalam posisi atau penyisipan elemen visual seperti logo terhadap huruf, yang seharusnya tertata rapi dalam dokumen asli.

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import cv2
import numpy as np

class HurufLogoAnalyzer:
    def __init__(self, root):
        self.root = root
        self.root.title("Hitung Persentase Huruf Tertimpa Logo")

        self.zoom_level = 1.0

        self.canvas_frame = tk.Frame(root)
        self.canvas_frame.pack(fill=tk.BOTH, expand=True)

        self.canvas = tk.Canvas(self.canvas_frame, bg="white",
                                cursor="cross")
        self.canvas.grid(row=0, column=0, sticky="nsew")

        self.v_scroll = tk.Scrollbar(self.canvas_frame,
                                     orient=tk.VERTICAL, command=self.canvas.yview)
```

```

        self.h_scroll = tk.Scrollbar(self.canvas_frame,
orient=tk.HORIZONTAL, command=self.canvas.xview)
        self.v_scroll.grid(row=0, column=1, sticky="ns")
        self.h_scroll.grid(row=1, column=0, sticky="ew")

        self.canvas.config(yscrollcommand=self.v_scroll.set,
xscrollcommand=self.h_scroll.set)
        self.canvas_frame.grid_rowconfigure(0, weight=1)
        self.canvas_frame.grid_columnconfigure(0, weight=1)

        self.btn_frame = tk.Frame(root)
        self.btn_frame.pack(pady=5)

        self.select_btn = tk.Button(self.btn_frame, text="Pilih Gambar",
command=self.select_image)
        self.select_btn.pack(side=tk.LEFT, padx=5)

        self.result_label = tk.Label(self.btn_frame, text="Persentase
tertinggi: -")
        self.result_label.pack(side=tk.LEFT, padx=10)

        self.new_instance_btn = tk.Button(self.btn_frame, text="New
Instance", command=self.create_new_instance)
        self.new_instance_btn.pack(side=tk.LEFT, padx=5)

        self.image_path = None
        self.cv_image = None
        self.tk_image = None
        self.start_x = self.start_y = self.end_x = self.end_y = 0
        self.rect = None
        self.displayed_image = None

        self.canvas.bind("<ButtonPress-1>", self.start_crop)
        self.canvas.bind("<B1-Motion>", self.update_crop)
        self.canvas.bind("<ButtonRelease-1>", self.finish_crop)
        self.canvas.bind("<MouseWheel>", self.zoom)

    def select_image(self):
        file_path = filedialog.askopenfilename(
            title="Pilih Gambar",
            filetypes=[("Image files", "*.jpg *.jpeg *.png")]
        )
        if not file_path:
            return
        self.image_path = file_path
        self.zoom_level = 1.0
        self.load_image()

    def load_image(self):
        self.cv_image = cv2.imread(self.image_path)
        self.display_image()

    def display_image(self):
        resized = cv2.resize(self.cv_image, None, fx=self.zoom_level,
fy=self.zoom_level, interpolation=cv2.INTER_LINEAR)
        image = cv2.cvtColor(resized, cv2.COLOR_BGR2RGB)
        pil_image = Image.fromarray(image)

```

```

        self.tk_image = ImageTk.PhotoImage(pil_image)

        self.canvas.delete("all")
        self.displayed_image = self.canvas.create_image(0, 0,
anchor=tk.NW, image=self.tk_image)
        self.canvas.config(scrollregion=self.canvas.bbox(tk.ALL))

    def start_crop(self, event):
        if not self.image_path:
            return
        self.start_x = self.canvas.canvasx(event.x)
        self.start_y = self.canvas.canvasy(event.y)
        if self.rect:
            self.canvas.delete(self.rect)
        self.rect = self.canvas.create_rectangle(
            self.start_x, self.start_y, self.start_x, self.start_y,
            outline="red", width=3
        )

    def update_crop(self, event):
        if not self.image_path:
            return
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.canvas.coords(self.rect, self.start_x, self.start_y,
self.end_x, self.end_y)

    def finish_crop(self, event):
        if not self.image_path:
            return
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.crop_and_analyze()

    def crop_and_analyze(self):
        scale = self.zoom_level
        x1 = int(min(self.start_x, self.end_x) / scale)
        y1 = int(min(self.start_y, self.end_y) / scale)
        x2 = int(max(self.start_x, self.end_x) / scale)
        y2 = int(max(self.start_y, self.end_y) / scale)

        cropped = self.cv_image[y1:y2, x1:x2]
        if cropped.size == 0:
            messagebox.showerror("Error", "Area crop tidak valid.")
            return

        hsv = cv2.cvtColor(cropped, cv2.COLOR_BGR2HSV)

        lower_black = np.array([0, 0, 0])
        upper_black = np.array([180, 255, 60])
        mask_huruf = cv2.inRange(hsv, lower_black, upper_black)

        lower_brown = np.array([20, 100, 30])
        upper_brown = np.array([100, 255, 200])
        mask_logo = cv2.inRange(hsv, lower_brown, upper_brown)

        total_huruf_pixels = cv2.countNonZero(mask_huruf)

```

```

if total_huruf_pixels == 0:
    self.result_label.config(text="Huruf tidak terdeteksi.")
    return

overlap = cv2.bitwise_and(mask_huruf, mask_logo)
overlap_pixels = cv2.countNonZero(overlap)

persentase = (overlap_pixels / total_huruf_pixels) * 100
self.result_label.config(
    text=f"Persentase OVERLAPPING: {persentase:.2f}%",
    font=("Arial", 30, "bold")
)

self.show_result_window(cropped, persentase)

def show_result_window(self, cropped_img, percentage):
    result_win = tk.Toplevel(self.root)
    result_win.title("Hasil Crop dan Persentase")

    # Hitung ukuran baru: tinggi tetap 640px, lebar proporsional
    h, w = cropped_img.shape[:2]
    new_height = 480
    aspect_ratio = w / h
    new_width = int(aspect_ratio * new_height)

    resized_crop = cv2.resize(cropped_img, (new_width, new_height),
                              interpolation=cv2.INTER_LINEAR)

    cropped_rgb = cv2.cvtColor(resized_crop, cv2.COLOR_BGR2RGB)
    pil_cropped = Image.fromarray(cropped_rgb)
    tk_cropped = ImageTk.PhotoImage(pil_cropped)

    label_img = tk.Label(result_win, image=tk_cropped)
    label_img.image = tk_cropped # simpan referensi agar tidak
    dihapus oleh Python GC
    label_img.pack(padx=10, pady=10)

    label_percent = tk.Label(
        result_win,
        text=f"Persentase OVERLAPPING: {percentage:.2f}%",
        font=("Arial", 20, "bold"),
        fg="blue"
    )
    label_percent.pack(pady=5)

def zoom(self, event):
    if self.cv_image is not None:
        if event.delta > 0:
            self.zoom_level *= 1.1
        elif event.delta < 0:
            self.zoom_level /= 1.1
        self.zoom_level = max(0.1, min(self.zoom_level, 10))
        self.display_image()

def create_new_instance(self):
    new_window = tk.Toplevel(self.root)

```

```

        HurufLogoAnalyzer(new_window)

if __name__ == "__main__":
    root = tk.Tk()
    app = HurufLogoAnalyzer(root)
    root.state('zoomed')
    root.mainloop()

```

Program di atas adalah sebuah sistem GUI berbasis Python dengan Tkinter yang dirancang untuk menganalisis berapa persen huruf dalam gambar yang tertimpa oleh logo. Metode ini sangat berguna dalam konteks forensik dokumen, terutama untuk mendeteksi keaslian ijazah, sertifikat, atau dokumen resmi yang sering memiliki logo stempel yang menimpa sebagian teks. Program ini tidak hanya menampilkan gambar dan memperbolehkan pengguna melakukan zoom, tetapi juga memberikan hasil dalam bentuk persentase overlap huruf yang terhalangi oleh logo.

Metode ini diawali dengan pemilihan gambar oleh pengguna melalui antarmuka grafis. Gambar kemudian dimuat menggunakan OpenCV dan ditampilkan di kanvas Tkinter. Gambar ini dapat di-zoom in atau zoom out dengan menggunakan scroll mouse, berkat dukungan fungsi `zoom()` yang mengatur level pembesaran gambar.

Setelah gambar ditampilkan, pengguna dapat memilih area yang ingin dianalisis dengan cara klik dan tarik mouse untuk melakukan cropping. Sistem mencatat koordinat awal dan akhir dari gerakan kursor tersebut, lalu menghitung ulang koordinat asli berdasarkan level zoom agar cropping dilakukan secara presisi terhadap citra asli, bukan hanya tampilan yang telah diperbesar.

Setelah area tertentu dicrop, area tersebut dikonversi dari format BGR ke HSV menggunakan `cv2.cvtColor()`. Konversi ini penting karena ruang warna HSV lebih memudahkan segmentasi objek berdasarkan warna, terutama ketika ingin membedakan huruf (yang biasanya hitam) dari stempel/logo (yang bisa berwarna coklat, merah, atau warna lain).

Untuk mendeteksi huruf, digunakan threshold HSV yang mengarah pada warna hitam dengan `cv2.inRange()` yang menghasilkan mask biner. Demikian pula, logo dideteksi dengan rentang warna HSV yang mengarah pada warna coklat. Dua mask ini menjadi representasi biner lokasi piksel huruf dan logo di area yang dianalisis.

Setelah kedua mask tersebut terbentuk, dilakukan operasi logika `bitwise_and()` untuk menentukan daerah mana dari huruf yang beririsan dengan logo. Operasi ini menghasilkan citra biner baru yang mewakili semua piksel huruf yang juga masuk ke dalam area warna logo—dengan kata lain, huruf yang tertimpa.

Langkah berikutnya adalah menghitung jumlah piksel huruf menggunakan `cv2.countNonZero()` dan menghitung juga jumlah piksel huruf yang tertimpa logo. Dengan dua nilai ini, dihitunglah persentase overlap dengan rumus:

$(\text{jumlah piksel tertimpa} / \text{total piksel huruf}) * 100.$

Hasil persentase ini kemudian ditampilkan dalam GUI pada label dengan ukuran font besar, memudahkan pengguna untuk segera melihat hasilnya. Selain itu, sebuah jendela baru akan muncul yang menampilkan area hasil cropping beserta hasil persentasenya. Ini sangat membantu dalam validasi visual terhadap hasil perhitungan.

Selain perhitungan, fungsi `show_result_window()` menampilkan area cropping yang telah diubah ukurannya agar tidak terlalu besar untuk layar, tetapi tetap menjaga rasio aspek aslinya. Konversi ke format PIL dan tampilan dengan `ImageTk.PhotoImage` digunakan karena Tkinter hanya menerima gambar dalam format tersebut.

Pengguna dapat mengulangi proses ini dengan memilih area lain di gambar yang sama, atau membuka gambar baru melalui tombol “New Instance”. Ini menciptakan jendela baru dengan instance baru dari GUI tanpa perlu memulai ulang aplikasi utama.

Program ini juga menangani kesalahan seperti cropping area kosong (misalnya jika pengguna tidak menyeret mouse) dengan `messagebox.showerror()`, menjaga agar aplikasi tetap ramah pengguna dan tahan terhadap kesalahan input.

Zoom pada gambar dikendalikan dengan event `<MouseWheel>` yang di-bind ke fungsi `zoom()`. Fungsi ini mengatur faktor pembesaran dan kemudian memanggil ulang `display_image()` untuk memperbarui tampilan gambar. Dengan batas bawah dan atas zoom, pengguna dapat memperbesar area spesifik hingga 10x untuk detail atau memperkecil hingga 0.1x.

Canvas dilengkapi dengan scrollbar horizontal dan vertikal, sehingga pengguna dapat menjelajahi gambar dengan mudah bahkan ketika telah diperbesar. Semua ini dibungkus dalam desain GUI berbasis grid untuk penataan yang rapi dan adaptif terhadap ukuran jendela.

Dari sisi teknis, program memanfaatkan keunggulan OpenCV dalam analisis citra, PIL untuk manipulasi gambar di Tkinter, dan Tkinter untuk antarmuka GUI. Ketiganya berkolaborasi untuk menciptakan sistem analisis visual yang cepat dan interaktif.

Dalam konteks dunia nyata, sistem ini bisa digunakan untuk menilai keaslian dokumen yang dicetak dengan stempel digital atau manual. Jika persentase overlap terlalu rendah, bisa menandakan stempel ditempelkan belakangan atau bahkan palsu, karena dalam dokumen asli logo biasanya menempa sebagian teks secara konsisten.

Metode segmentasi berdasarkan HSV juga memberikan fleksibilitas untuk mendeteksi warna logo lain, tidak hanya coklat. Ini bisa dimodifikasi untuk mendeteksi merah, biru, atau warna lain tergantung jenis dokumen yang dianalisis. Artinya, sistem ini adaptif dan dapat dikustomisasi untuk kebutuhan analisis forensik visual yang berbeda-beda.

Dengan pendekatan GUI ini, sistem menjadi sangat user-friendly, memungkinkan penggunaan oleh orang yang tidak terlalu paham teknis pemrograman, seperti petugas arsip, dosen, atau penyidik dokumen. Mereka hanya perlu memilih gambar, crop area yang diinginkan, dan sistem langsung menghitung dan menampilkan hasilnya.

Secara keseluruhan, metode dalam program ini merupakan kombinasi yang efisien dari analisis warna, deteksi objek biner, dan antarmuka grafis. Ini menjadikannya alat yang kuat dalam mendeteksi potensi pemalsuan atau manipulasi dokumen visual dengan cara yang interaktif dan mudah digunakan.

HASIL IMPLEMENTASI OVERLAPPING DETECTION PADA RUANG WARNA HSV

Berikut penjelasan hasil implementasi overlapping detection antara dua dokumen ijazah—yakni atas nama Frono Jiwo dan Joko Widodo—dengan asumsi bahwa ijazah atas nama Joko Widodo adalah yang diduga palsu, berdasarkan analisis tumpang tindih cap logo terhadap huruf dalam kata “IJAZAH” menggunakan ruang warna HSV.

Pada analisis ini digunakan metode pemisahan warna dalam ruang HSV yang efektif dalam mendeteksi objek berdasarkan rona (hue), kejenuhan (saturation), dan intensitas cahaya (value). Huruf-huruf dalam dokumen cenderung berwarna hitam, sedangkan cap logo berwarna coklat kemerahan. Dengan rentang warna yang didefinisikan dalam HSV, dibuat dua masker berbeda: satu untuk huruf dan satu untuk logo. Kemudian dilakukan operasi bitwise untuk mengukur bagian huruf yang tertimpa logo.

Berdasarkan hasil crop dan persentase overlapping yang ditampilkan, dokumen atas nama Frono Jiwo memperlihatkan konsistensi alami dari proses pencetakan, di mana cap logo menimpa sebagian huruf. Misalnya, huruf “A” tertimpa 21.01%, “I” sekitar 14.71%, dan “E” hingga 18.02%. Nilai-nilai ini mengindikasikan adanya pencetakan logo yang dilakukan di

atas huruf sebagaimana lazim dilakukan dalam prosedur penerbitan resmi dokumen ijazah.



Sebaliknya, dokumen atas nama Joko Widodo menunjukkan nilai overlapping yang sangat rendah untuk huruf-huruf yang berada tepat di area logo. Misalnya, huruf "A" pada kata IJAZAH hanya menunjukkan tumpang tindih sebesar 1.00%, "E" hanya 4.48%, dan "D" sebesar 5.95%. Nilai ini sangat janggal karena secara visual logo berada sangat dekat dengan huruf, namun justru tidak menunjukkan adanya tumpang tindih berarti secara piksel. Ini memberikan indikasi bahwa logo bisa saja

ditambahkan setelah teks dicetak, atau bahkan ditambahkan secara digital.

Iniilah yang menjadi titik kritis dalam analisis: logo pada ijazah Joko Widodo tampak “berhati-hati” menghindari huruf, seolah-olah tidak mencetak secara natural di atas teks. Hal ini bertolak belakang dengan dokumen Frono Jiwo di mana overlapping terjadi secara merata dan natural tanpa terkesan diatur. Overlapping alami pada dokumen asli seharusnya menimpa huruf sebagian, menciptakan nilai deteksi antara 10% hingga 25%, tergantung pada bentuk huruf dan tekanan cap.

Dalam forensik digital, ketidakwajaran pada overlapping seperti ini dapat menjadi indikator penting pemalsuan. Jika logo secara visual tampak berada di atas huruf tetapi tidak memiliki nilai overlapping yang signifikan, maka ada kemungkinan proses pencapannya tidak mengikuti prosedur resmi. Pada kasus dokumen Joko Widodo, overlapping rata-rata hanya berkisar antara 1% hingga 5%, jauh dari batas wajar dokumen asli yang mencetak logo setelah teks.

Teknik ini memberikan pendekatan ilmiah dalam membedakan dokumen asli dan yang diduga palsu, dengan mengukur secara kuantitatif seberapa banyak piksel huruf tertutupi oleh logo. Dengan hasil yang ditampilkan dalam GUI berbasis Tkinter, pengguna dapat memilih huruf mana yang ingin diperiksa, melihat hasil crop, serta membaca persentase overlapping secara real time, tanpa interpretasi subjektif.

Fitur zoom dan scrollbar memperkuat aspek presisi analisis ini karena memungkinkan pengguna mengamati hingga tingkat piksel. Dalam analisis manual, perbedaan halus ini sulit ditangkap mata manusia. Tetapi dengan ruang warna HSV, perbedaan warna, bahkan dalam kondisi pencahayaan berbeda, tetap dapat terdeteksi dengan akurat.

Keunggulan metode ini juga terletak pada kemampuannya diulang dan diverifikasi. Siapa pun dengan akses terhadap dokumen dan alat yang sama dapat melakukan analisis ulang dan mendapatkan hasil serupa. Ini sesuai dengan prinsip transparansi dalam digital forensics.

Lebih lanjut, dokumen yang diduga palsu sering kali menunjukkan upaya untuk membuat tampilannya “terlihat otentik”, namun justru meninggalkan jejak tidak alami seperti tumpang tindih huruf dan logo yang tidak konsisten. Dalam kasus ini, ijazah Joko Widodo justru memperlihatkan tanda-tanda tersebut, sedangkan ijazah Frono Jiwo tampil lebih alami dan meyakinkan dari sisi proses pencetakan dan integritas visual.

Penggunaan ruang warna HSV juga memungkinkan algoritma ini bekerja meski latar belakang dokumen bervariasi. Ini menjadikan metode tersebut lebih tahan terhadap variasi visual seperti penuaan dokumen atau gangguan pencahayaan pada foto. Oleh karena itu, deteksi overlapping berdasarkan HSV adalah salah satu pendekatan paling stabil untuk forensik digital dokumen.

Bila digunakan secara luas, teknik ini dapat digunakan dalam pengujian keaslian berbagai dokumen penting lain, seperti sertifikat, transkrip, atau akta lahir. Selama dokumen memiliki elemen teks dan cap atau logo, maka deteksi overlapping dapat diterapkan secara langsung dan efektif.

Dalam konteks kasus ini, hasil analisis memperkuat dugaan bahwa ijazah Joko Widodo mungkin bukan hasil cetakan resmi universitas, karena menunjukkan pola overlapping yang tidak biasa, konsisten rendah, dan terkesan direkayasa. Sebaliknya, ijazah Frono Jiwo menunjukkan hasil overlapping yang sesuai dengan karakteristik dokumen resmi yang dicetak secara sah.

Sebagai kesimpulan, analisis overlapping berbasis HSV pada dua dokumen ini memperlihatkan adanya anomali yang signifikan pada dokumen Joko Widodo yang berpotensi mengindikasikan pemalsuan. Sementara dokumen Frono Jiwo tampak selaras dengan karakteristik pencetakan otentik, tumpang tindih yang lebih merata dan realistis mendukung kesahihan dokumen tersebut dari sudut pandang forensik digital.

IMPLEMENTASI OVERLAPPING DETECTION DENGAN ADAPTIVE THRESHOLDING

Teknik overlapping detection yang diterapkan dalam program ini merupakan pendekatan analisis citra digital untuk menghitung seberapa besar area huruf yang tertimpa oleh logo dalam suatu dokumen, seperti ijazah atau sertifikat. Tujuannya adalah untuk mengukur tingkat interseksi visual antara dua objek berbeda dalam gambar, yakni teks dan elemen grafis (logo atau cap stempel). Teknik ini sangat penting dalam bidang forensik digital, khususnya untuk menguji keaslian dokumen dan mendeteksi kemungkinan manipulasi atau pemalsuan.

Langkah pertama yang dilakukan dalam metode ini adalah memungkinkan pengguna memilih citra (gambar) dokumen yang akan dianalisis melalui antarmuka pengguna grafis (GUI) berbasis Tkinter. Setelah gambar dipilih, pengguna dapat memperbesar atau memperkecil tampilan dan memilih area tertentu dari gambar yang akan dianalisis. Hal ini memberikan fleksibilitas tinggi kepada pengguna untuk memfokuskan analisis pada bagian yang dianggap relevan, seperti huruf yang berada di bawah cap stempel atau logo institusi.

Setelah area tertentu dicrop, proses analisis dimulai dengan mengubah citra ke dalam ruang warna grayscale. Grayscale merupakan representasi satu kanal yang sangat berguna dalam mendeteksi elemen teks (yang umumnya berwarna gelap, seperti hitam) karena perbedaan kontrasnya yang tinggi terhadap latar belakang. Di ruang grayscale ini, algoritma menerapkan adaptive thresholding, yakni teknik binarisasi adaptif untuk memisahkan huruf dari latar belakang berdasarkan perbedaan intensitas lokal. Teknik ini lebih unggul dibanding thresholding global pada gambar dengan pencahayaan tidak seragam.

Adaptive thresholding digunakan dalam mode Gaussian, yang mempertimbangkan rata-rata berbobot piksel sekitar untuk menetapkan ambang batas. Hasilnya adalah citra biner di mana piksel putih mewakili huruf dan piksel hitam mewakili latar belakang. Ini membentuk masker huruf (thresh_huruf) yang akan menjadi referensi utama dalam menghitung sebaran teks dalam gambar.

Langkah selanjutnya adalah memproses gambar yang sama ke dalam ruang warna LAB. LAB color space dipilih karena menawarkan kestabilan warna yang lebih tinggi dibanding RGB dan HSV, terutama dalam mendeteksi warna spesifik seperti coklat, emas, atau merah tua yang umum digunakan dalam stempel atau logo dokumen resmi. Dalam ruang LAB, nilai-nilai kanal L (luminansi) dan kanal A/B (warna hijau-merah dan biru-kuning) dapat digunakan untuk membentuk batas bawah dan atas dari warna logo.

Dengan menerapkan `cv2.inRange()` terhadap gambar dalam LAB, program menghasilkan masker logo—citra biner yang merepresentasikan area logo. Area ini kemudian diperbaiki dengan morphological closing, yaitu serangkaian operasi dilatasi dan erosi menggunakan kernel persegi. Tujuannya adalah menutup celah kecil atau lubang di dalam area logo sehingga hasil segmentasi menjadi lebih solid dan akurat.

Dua masker yang telah terbentuk—yakni huruf (`thresh_huruf`) dan logo (`mask_logo`)—kemudian dikombinasikan menggunakan operasi bitwise AND. Operasi ini menghasilkan citra biner baru (`overlap`) yang menandai piksel yang termasuk dalam kedua kategori sekaligus, yaitu bagian huruf yang tertimpa oleh logo. Dengan menghitung jumlah piksel putih pada `overlap`, sistem memperoleh data kuantitatif tentang luas interseksi antara huruf dan logo.

Untuk menghitung persentase overlapping, jumlah piksel `overlap` dibagi dengan jumlah total piksel dalam masker huruf (`thresh_huruf`), lalu dikalikan dengan 100. Nilai persentase ini merepresentasikan seberapa besar bagian huruf yang tertutupi oleh logo, dan ditampilkan dalam GUI sebagai indikator visual untuk pengguna.

Hasil crop dan nilai persentase overlapping kemudian ditampilkan dalam jendela terpisah yang menampilkan visual dari area yang dianalisis, serta teks berformat besar yang menampilkan nilai persentase dengan jelas. Visualisasi ini sangat penting untuk menyampaikan bukti visual kepada

pengguna mengenai bagian huruf yang mungkin telah tertimpa logo secara otentik atau mungkin dimanipulasi.

Keakuratan deteksi overlap sangat bergantung pada kualitas citra, kontras warna antar objek, dan konsistensi visual dokumen. Oleh karena itu, penggunaan adaptive thresholding dan ruang warna LAB secara bersamaan memberikan keseimbangan antara sensitivitas dan ketahanan terhadap gangguan seperti noise atau pencahayaan tidak merata.

Metode ini juga menyediakan fitur zoom dan scrolling, sehingga pengguna dapat dengan mudah menavigasi dan menganalisis gambar beresolusi tinggi. Fleksibilitas ini menjadikan GUI sebagai alat bantu yang kuat dalam forensik visual, terutama dalam investigasi otentikasi dokumen hukum atau administratif.

Dalam konteks verifikasi ijazah atau dokumen resmi lainnya, kehadiran huruf yang sepenuhnya tidak tertimpa logo dapat menjadi indikasi bahwa teks tersebut ditambahkan secara digital setelah stempel atau logo ditambahkan. Sebaliknya, jika sebagian besar area huruf tertimpa oleh logo, hal ini lebih cenderung mendukung keaslian dokumen karena menunjukkan proses pencetakan yang koheren dan berurutan.

Aplikasi ini juga mendukung analisis perbandingan, yakni pengguna dapat membuka lebih dari satu jendela dan menganalisis beberapa huruf dari dokumen berbeda untuk melihat pola atau anomali. Ini memungkinkan analisis bertingkat, misalnya untuk membandingkan dokumen dari dua sumber atau meneliti keaslian dokumen yang dicurigai palsu.

Teknik overlapping detection seperti ini sangat relevan dalam kasus pemalsuan dokumen digital maupun cetak, karena manipulasi digital seringkali meninggalkan jejak visual yang dapat dianalisis secara statistik dan visual. Dengan pengukuran kuantitatif, kesimpulan tidak hanya bersifat subjektif, tetapi didukung oleh metrik numerik yang dapat diverifikasi ulang.

Penggunaan ruang LAB dibandingkan HSV juga merupakan peningkatan signifikan dalam deteksi warna logo karena HSV memiliki kelemahan dalam kondisi pencahayaan bervariasi, sedangkan LAB lebih stabil terhadap perubahan intensitas dan saturasi.

Meskipun metode ini cukup akurat, masih terdapat ruang untuk peningkatan seperti menambahkan modul OCR untuk mendeteksi karakter secara otomatis, menambahkan fitur heatmap overlap, atau menggunakan machine learning untuk klasifikasi otomatis tingkat otentisitas dokumen.

Dalam pengembangan selanjutnya, teknik ini juga dapat dikombinasikan dengan metode lain seperti template matching, fitur SIFT/SURF, atau analisis tekstur untuk memperkaya deteksi kegagalan pada dokumen. Hal ini akan meningkatkan kekuatan forensik visual dalam membongkar dokumen yang telah dimanipulasi secara halus.

Secara keseluruhan, teknik overlapping detection berbasis adaptive thresholding dan segmentasi warna dalam ruang LAB adalah pendekatan yang kuat, akurat, dan aplikatif dalam ranah forensik dokumen. Metode ini tidak hanya mengidentifikasi tumpang tindih secara visual, tetapi juga menyajikan metrik numerik yang dapat digunakan untuk mendukung investigasi dan pengambilan keputusan hukum atau administratif.

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import cv2
import numpy as np

class HurufLogoAnalyzer:
    def __init__(self, root):
        self.root = root
        self.root.title("Hitung Persentase Huruf Tertimpa Logo")

        self.zoom_level = 1.0

        self.canvas_frame = tk.Frame(root)
        self.canvas_frame.pack(fill=tk.BOTH, expand=True)

        self.canvas = tk.Canvas(self.canvas_frame, bg="white",
                                cursor="cross")
        self.canvas.grid(row=0, column=0, sticky="nsew")
```



```

        self.v_scroll = tk.Scrollbar(self.canvas_frame,
orient=tk.VERTICAL, command=self.canvas.yview)
        self.h_scroll = tk.Scrollbar(self.canvas_frame,
orient=tk.HORIZONTAL, command=self.canvas.xview)
        self.v_scroll.grid(row=0, column=1, sticky="ns")
        self.h_scroll.grid(row=1, column=0, sticky="ew")

        self.canvas.config(yscrollcommand=self.v_scroll.set,
xscrollcommand=self.h_scroll.set)
        self.canvas_frame.grid_rowconfigure(0, weight=1)
        self.canvas_frame.grid_columnconfigure(0, weight=1)

        self.btn_frame = tk.Frame(root)
        self.btn_frame.pack(pady=5)

        self.select_btn = tk.Button(self.btn_frame, text="Pilih Gambar",
command=self.select_image)
        self.select_btn.pack(side=tk.LEFT, padx=5)

        self.result_label = tk.Label(self.btn_frame, text="Persentase
tertempa: -")
        self.result_label.pack(side=tk.LEFT, padx=10)

        self.new_instance_btn = tk.Button(self.btn_frame, text="New
Instance", command=self.create_new_instance)
        self.new_instance_btn.pack(side=tk.LEFT, padx=5)

        self.image_path = None
        self.cv_image = None
        self.tk_image = None
        self.start_x = self.start_y = self.end_x = self.end_y = 0
        self.rect = None
        self.displayed_image = None

        self.canvas.bind("<ButtonPress-1>", self.start_crop)
        self.canvas.bind("<B1-Motion>", self.update_crop)
        self.canvas.bind("<ButtonRelease-1>", self.finish_crop)
        self.canvas.bind("<MouseWheel>", self.zoom)

    def select_image(self):
        file_path = filedialog.askopenfilename(
            title="Pilih Gambar",
            filetypes=[("Image files", "*.jpg *.jpeg *.png")]
        )
        if not file_path:
            return
        self.image_path = file_path
        self.zoom_level = 1.0
        self.load_image()

    def load_image(self):
        self.cv_image = cv2.imread(self.image_path)
        self.display_image()

    def display_image(self):
        resized = cv2.resize(self.cv_image, None, fx=self.zoom_level,
fy=self.zoom_level, interpolation=cv2.INTER_LINEAR)

```

```

        image = cv2.cvtColor(resized, cv2.COLOR_BGR2RGB)
        pil_image = Image.fromarray(image)
        self.tk_image = ImageTk.PhotoImage(pil_image)

        self.canvas.delete("all")
        self.displayed_image = self.canvas.create_image(0, 0,
anchor=tk.NW, image=self.tk_image)
        self.canvas.config(scrollregion=self.canvas.bbox(tk.ALL))

    def start_crop(self, event):
        if not self.image_path:
            return
        self.start_x = self.canvas.canvasx(event.x)
        self.start_y = self.canvas.canvasy(event.y)
        if self.rect:
            self.canvas.delete(self.rect)
        self.rect = self.canvas.create_rectangle(
            self.start_x, self.start_y, self.start_x, self.start_y,
            outline="red", width=3
        )

    def update_crop(self, event):
        if not self.image_path:
            return
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.canvas.coords(self.rect, self.start_x, self.start_y,
self.end_x, self.end_y)

    def finish_crop(self, event):
        if not self.image_path:
            return
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.crop_and_analyze()

    def crop_and_analyze(self):
        scale = self.zoom_level
        x1 = int(min(self.start_x, self.end_x) / scale)
        y1 = int(min(self.start_y, self.end_y) / scale)
        x2 = int(max(self.start_x, self.end_x) / scale)
        y2 = int(max(self.start_y, self.end_y) / scale)

        cropped = self.cv_image[y1:y2, x1:x2]
        if cropped.size == 0:
            messagebox.showerror("Error", "Area crop tidak valid.")
            return

        # Huruf (teks hitam) menggunakan adaptive threshold
        gray = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
        thresh_huruf = cv2.adaptiveThreshold(
            gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
            cv2.THRESH_BINARY_INV, 11, 3
        )

        # Logo menggunakan ruang warna LAB
        lab = cv2.cvtColor(cropped, cv2.COLOR_BGR2LAB)

```

```

lower_logo = np.array([120, 130, 135])
upper_logo = np.array([200, 170, 180])
mask_logo = cv2.inRange(lab, lower_logo, upper_logo)

# Morfologi untuk perbaikan area logo
kernel = np.ones((3, 3), np.uint8)
mask_logo = cv2.morphologyEx(mask_logo, cv2.MORPH_CLOSE, kernel,
iterations=2)

total_huruf_pixels = cv2.countNonZero(thresh_huruf)
if total_huruf_pixels == 0:
    self.result_label.config(text="Huruf tidak terdeteksi.")
    return

overlap = cv2.bitwise_and(thresh_huruf, mask_logo)
overlap_pixels = cv2.countNonZero(overlap)

persentase = (overlap_pixels / total_huruf_pixels) * 100
self.result_label.config(
    text=f"Persentase OVERLAPPING: {persentase:.2f}%",
    font=("Arial", 30, "bold")
)

self.show_result_window(cropped, persentase)

def show_result_window(self, cropped_img, percentage):
    result_win = tk.Toplevel(self.root)
    result_win.title("Hasil Crop dan Persentase")

    h, w = cropped_img.shape[:2]
    new_height = 480
    aspect_ratio = w / h
    new_width = int(aspect_ratio * new_height)

    resized_crop = cv2.resize(cropped_img, (new_width, new_height),
interpolation=cv2.INTER_LINEAR)
    cropped_rgb = cv2.cvtColor(resized_crop, cv2.COLOR_BGR2RGB)
    pil_cropped = Image.fromarray(cropped_rgb)
    tk_cropped = ImageTk.PhotoImage(pil_cropped)

    label_img = tk.Label(result_win, image=tk_cropped)
    label_img.image = tk_cropped
    label_img.pack(padx=10, pady=10)

    label_percent = tk.Label(
        result_win,
        text=f"Persentase OVERLAPPING: {percentage:.2f}%",
        font=("Arial", 20, "bold"),
        fg="blue"
    )
    label_percent.pack(pady=5)

def zoom(self, event):
    if self.cv_image is not None:
        if event.delta > 0:
            self.zoom_level *= 1.1
        elif event.delta < 0:

```

```

        self.zoom_level /= 1.1
        self.zoom_level = max(0.1, min(self.zoom_level, 10))
        self.display_image()

    def create_new_instance(self):
        new_window = tk.Toplevel(self.root)
        HurufLogoAnalyzer(new_window)

if __name__ == "__main__":
    root = tk.Tk()
    app = HurufLogoAnalyzer(root)
    root.state('zoomed')
    root.mainloop()

```

HASIL IMPLEMENTASI OVERLAPPING DETECTION DENGAN ADAPTIVE THRESHOLDING

Berikut adalah analisis hasil overlapping detection berbasis adaptive thresholding dari dua dokumen ijazah yang dibandingkan—yang satu diduga milik Frono Jiwo (asli) dan yang lain atas nama Joko Widodo (diduga palsu)—dengan menyoroti huruf-huruf yang terkena logo di masing-masing dokumen:

Hasil deteksi pada huruf “A” pertama dari ijazah Frono menunjukkan nilai persentase overlapping sebesar 29.48%. Angka ini sangat tinggi dan mengindikasikan bahwa bagian huruf “A” berada tepat di atas bagian logo yang cukup pekat, sehingga sebagian besar bagian hitam dari huruf tertimpa oleh warna khas logo, mendukung bahwa logo dicetak setelah huruf dicetak.

Sebaliknya, huruf “A” pertama dari ijazah atas nama Joko Widodo hanya memiliki persentase overlapping 3.59%. Ini menunjukkan bahwa hampir tidak ada bagian dari huruf yang bersinggungan dengan area logo yang terdeteksi, seolah huruf ditambahkan setelah logo, yang mencurigakan dalam konteks keaslian dokumen.

Huruf “O” dari ijazah Frono memiliki nilai overlapping sebesar 16.74%, yang berarti sebagian besar tepi luar huruf beririsan dengan bagian tengah logo. Ini menunjukkan keselarasan antara cetakan teks dan

elemen logo, menunjukkan koherensi desain yang tipikal pada dokumen asli.

Huruf “O” pada dokumen Joko Widodo, dengan hanya 2.80% overlapping, sangat minim tertimpa oleh logo. Hal ini bisa mengindikasikan bahwa huruf mungkin ditempel atau ditambahkan secara digital setelah logo dicetak, yang bisa menjadi indikasi adanya modifikasi.



Huruf “I” dari ijazah Frano mencatat nilai tertinggi dalam analisis ini, yaitu sebesar 53.85%. Huruf ini secara dominan tumpang tindih dengan area padat dari logo, yang menandakan integrasi yang sangat kuat antara

huruf dan logo. Dalam dokumen asli, hal ini sering terjadi karena stempel dan teks dicetak secara berurutan dalam satu tahap produksi.

Sebaliknya, huruf “I” pada ijazah Joko Widodo hanya menunjukkan overlapping sebesar 15.47%. Meski angka ini tidak serendah huruf lain dalam dokumen tersebut, nilai ini masih jauh dari nilai pada dokumen Frono, dan menunjukkan bahwa sebagian besar huruf tidak tertutup oleh elemen logo.

Huruf “D” dari Frono tercatat dengan nilai 25.49%, nilai ini cukup tinggi dan mengindikasikan bahwa hampir seperempat bagian dari area huruf “D” tumpang tindih dengan warna dari logo. Warna dari logo yang khas terlihat menindih sebagian besar area huruf.

Sebaliknya, huruf “D” pada dokumen Joko Widodo menunjukkan nilai overlapping 7.75%. Ini menunjukkan bahwa huruf “D” sebagian besar tidak tersentuh oleh warna logo. Nilai ini menunjukkan kemungkinan ketidaksinkronan antara huruf dan elemen visual latar.

Huruf “E” dari Frono mencatat nilai overlapping sebesar 53.19%, hampir setara dengan huruf “I” sebelumnya. Hal ini mencerminkan konsistensi cetakan pada dokumen Frono yang memperlihatkan integrasi antara huruf dan logo. Nilai di atas 50% menandakan kemungkinan besar huruf ditulis lebih dulu sebelum logo distempel.

Huruf “E” dari Joko Widodo hanya memiliki nilai overlapping 0.50%, yang sangat kecil, bahkan bisa diabaikan. Nilai ini sangat mencurigakan dan menunjukkan bahwa huruf hampir sepenuhnya berada di atas stempel tanpa interaksi warna, yang tidak wajar untuk dokumen resmi.

Huruf “A” kedua dari Frono memiliki nilai 5.49%. Meski lebih kecil dibandingkan huruf lainnya, nilai ini tetap menunjukkan bahwa terdapat sebagian kecil dari huruf yang tertimpa oleh warna logo. Masih lebih kredibel dibandingkan nilai sangat kecil di dokumen Joko Widodo.

Huruf “A” pada dokumen Joko Widodo memiliki nilai overlapping hanya 2.34%. Sekali lagi, nilai yang terlalu kecil ini menegaskan pola umum dari

dokumen ini: minimnya tumpang tindih antara huruf dan logo, yang berpotensi mengindikasikan penambahan teks setelah logo ditambahkan.

Dalam konteks forensik digital, perbedaan ekstrem antara nilai overlapping dari dokumen Frono dan dokumen Joko Widodo sangat signifikan. Dokumen Frono menunjukkan konsistensi tinggi antara elemen teks dan logo, mendukung bahwa dokumen tersebut dicetak secara resmi dan tidak mengalami modifikasi digital.

Sebaliknya, dokumen Joko Widodo menunjukkan pola yang sebaliknya: nilai overlapping sangat rendah, dan sebagian besar huruf tidak tertindih oleh logo. Ini menandakan bahwa teks “IJAZAH” bisa saja ditambahkan secara digital atau dicetak ulang setelah stempel sudah ada.

Overlapping detection menggunakan adaptive thresholding memberikan keuntungan besar dalam mendeteksi huruf gelap pada latar terang, dengan menyesuaikan ambang secara lokal. Ini membuat segmentasi huruf lebih akurat bahkan jika pencahayaan gambar tidak merata.

Adaptive thresholding memberikan fleksibilitas terhadap variasi pencahayaan 136temp di dalam dokumen, dan menghasilkan deteksi huruf yang lebih solid dibandingkan dengan metode threshold global atau penggunaan ruang warna HSV semata.

Metode ini sangat sesuai untuk dokumen yang memiliki banyak tekstur dan warna, karena adaptive threshold dapat menyaring noise latar belakang dan menampilkan hanya huruf dengan akurasi tinggi, sebagai dasar untuk perhitungan overlapping.

Penggunaan ruang warna LAB untuk mendeteksi logo dalam dokumen juga memperkuat metode ini, karena LAB lebih stabil dalam memisahkan warna khas seperti coklat atau emas dari logo tanpa dipengaruhi pencahayaan yang berubah.

Analisis overlap ini bekerja dengan menghitung jumlah piksel putih (yang mewakili huruf) yang juga tertimpa oleh warna-warna logo. Operasi bitwise AND antara masker huruf dan logo memberikan hasil yang bersih dan terfokus pada bagian yang relevan.

Hasil crop dan zoom pada GUI memungkinkan verifikasi visual langsung oleh pengguna, sehingga hasil deteksi tidak hanya numerik tetapi juga dapat dipantau secara visual. Ini penting dalam forensik untuk menunjukkan bukti nyata pada pihak ketiga.

GUI yang menyertakan fitur new instance memungkinkan pemeriksa membuka beberapa area crop sekaligus dari gambar yang berbeda, mempercepat proses investigasi dokumen secara 137temple137 dan komparatif.

Kehadiran scrollbar dan zoom meningkatkan kemampuan inspeksi detail, memungkinkan pengguna 137temp pada tepi huruf atau logo untuk melihat apakah 137temple tindih itu nyata atau hanya artefak visual semata.

Secara keseluruhan, nilai overlapping yang tinggi pada dokumen Frono dan sangat rendah pada dokumen Joko Widodo menunjukkan inkonsistensi 137temple137c137. Dalam ranah hukum, hasil ini dapat menjadi salah satu bukti teknis untuk menguji keaslian dokumen.

Perbedaan antara nilai overlapping dalam dua dokumen ini tidak bisa dijelaskan dengan variasi cetak biasa. Secara 137temple137c dan visual, pola tersebut mengarah pada kesimpulan yang sangat berbeda terkait otentisitas keduanya.

Metode ini dapat diperluas ke berbagai jenis dokumen lain—akte lahir, surat keputusan, sertifikat akademik—yang menggunakan logo atau 137temple yang menutupi teks, menjadikan overlapping detection sebagai alat forensik digital yang krusial.

Implementasi yang sederhana namun efektif dari metode ini menjadikannya dapat diakses oleh penyidik, akademisi, maupun pemeriksa dokumen untuk melakukan investigasi awal secara mandiri sebelum dilakukan pengujian lanjutan di laboratorium forensik.

Akhirnya, berdasarkan hasil analisis ini, secara teknis terdapat pola yang kuat untuk menyatakan bahwa dokumen atas nama Joko Widodo sangat mencurigakan secara visual karena tidak menunjukkan karakteristik cetakan asli seperti halnya dokumen Frono Jiwo.

IMPLEMENTASI OVERLAPPING DETECTION DENGAN LBP (Local Binary Pattern)

Program berikut merupakan sebuah aplikasi antarmuka grafis (GUI) yang dibangun menggunakan Python dengan pustaka tkinter, OpenCV, dan PIL untuk mendeteksi tingkat penumpukan (overlapping) antara huruf dan logo dalam sebuah gambar. Aplikasi ini menggabungkan teknik pemrosesan citra menggunakan Local Binary Pattern (LBP) dan segmentasi warna di ruang warna LAB untuk melakukan analisis visual.

Ketika aplikasi dibuka, jendela utama GUI akan muncul dengan berbagai komponen antarmuka seperti kanvas untuk menampilkan gambar, scrollbar untuk navigasi, serta tombol-tombol seperti "Pilih Gambar", "New Instance", dan label untuk menampilkan hasil persentase overlapping. Tombol "Pilih Gambar" memungkinkan pengguna untuk memilih sebuah file gambar dari komputer mereka.

Gambar yang dipilih akan dimuat menggunakan OpenCV, lalu ditampilkan di dalam kanvas dengan konversi warna BGR ke RGB agar cocok dengan format tampilan ImageTk. Ukuran gambar bisa diubah secara dinamis menggunakan fungsi zoom dengan scroll mouse, yang mengatur faktor pembesaran `zoom_level`.

Pengguna dapat menyeleksi bagian tertentu dari gambar dengan mengklik dan menyeret mouse di atas kanvas. Ketika mouse dilepas,

koordinat awal dan akhir crop akan dihitung dan digunakan untuk mengambil bagian citra yang diinginkan.

Bagian citra yang telah dicrop akan diproses lebih lanjut untuk mendeteksi huruf dan area logo. Huruf-huruf akan diekstraksi menggunakan metode Local Binary Pattern (LBP), yang bekerja dengan membandingkan setiap piksel dengan tetangga sekitarnya dan membentuk pola biner yang kemudian dikonversi menjadi nilai desimal antara 0 sampai 255.

Hasil dari proses LBP kemudian dibinarisasi menggunakan metode threshold Otsu. Ini membantu untuk memisahkan area huruf dari latar belakang dalam bentuk peta biner, di mana area huruf akan muncul sebagai piksel putih (nilai 255) di atas latar belakang hitam (nilai 0).

Sementara itu, area logo dideteksi menggunakan segmentasi warna dalam ruang warna LAB. Nilai warna LAB dipilih karena sensitivitasnya yang tinggi terhadap perbedaan warna visual manusia, menjadikannya cocok untuk membedakan warna logo dari teks hitam.

Setelah mendapatkan masker biner untuk area logo menggunakan fungsi `cv2.inRange`, dilakukan operasi morfologi `morphologyEx` dengan kernel kecil berukuran 3x3 untuk menutup lubang-lubang kecil dalam area logo, memperhalus batasannya, dan membuatnya lebih kompak.

Setelah kedua area—huruf dan logo—terdeteksi, dilakukan operasi logika `bitwise_and` untuk menemukan bagian huruf yang tertimpa oleh logo. Piksel-piksel hasil tumpang tindih inilah yang menjadi fokus utama analisis persentase overlapping.

Program kemudian menghitung jumlah total piksel huruf (`total_huruf_pixels`) dan jumlah piksel yang tertimpa logo (`overlap_pixels`). Persentase overlapping dihitung dengan membagi jumlah piksel tertimpa dengan jumlah piksel huruf dan dikalikan 100 untuk menghasilkan persentase.

Hasil persentase tersebut ditampilkan secara langsung dalam GUI utama dan juga dalam sebuah jendela popup yang menampilkan gambar hasil crop serta informasi persentasenya. Tampilan ini membantu pengguna untuk secara visual meninjau area yang dipilih dan hasil perhitungan overlapping.

Fungsi `local_binary_pattern` yang digunakan di sini merupakan implementasi manual dari metode LBP. Ia memproses setiap piksel (kecuali tepi) dengan membandingkan intensitas pusat piksel dengan delapan tetangganya. Setiap perbandingan menghasilkan 1 atau 0 tergantung apakah tetangga lebih besar atau sama dengan pusat.

Setiap hasil perbandingan ini kemudian digeser ke posisi bit yang sesuai, dikombinasikan menggunakan operasi bitwise OR, dan disimpan sebagai nilai LBP untuk piksel tersebut. Hasil akhirnya adalah citra baru dengan intensitas berdasarkan pola tekstur lokal dari citra asli.

Metode ini sangat efektif untuk mengekstrak pola tekstur lokal seperti kontur huruf, karena tidak terlalu tergantung pada intensitas absolut, melainkan pada perubahan tekstur lokal di sekitar suatu piksel. Ini menjadikannya lebih robust terhadap perubahan pencahayaan.

Selain itu, aplikasi ini mendukung pembuatan instance baru dari GUI. Ini berguna bila pengguna ingin membandingkan lebih dari satu gambar sekaligus dalam beberapa jendela yang terpisah, tanpa perlu menutup aplikasi utama.

Fungsi zoom diimplementasikan untuk meningkatkan kenyamanan dalam menavigasi citra besar atau untuk memperhatikan detail kecil pada gambar. Ini dilakukan dengan menyesuaikan faktor zoom dan merespons event scroll mouse (`<MouseWheel>`).

Dengan menggunakan OpenCV untuk semua pemrosesan citra dan PIL untuk konversi ke format yang bisa ditampilkan di tkinter, aplikasi ini menjembatani antara teknologi pemrosesan citra dan antarmuka pengguna yang intuitif.

Keseluruhan struktur program bersifat modular, dengan pemisahan jelas antara komponen GUI dan pemrosesan data. Ini memudahkan pengembangan lebih lanjut, seperti menambahkan fitur penyimpanan hasil, anotasi, atau analisis LBP histogram.

Aplikasi ini cocok digunakan dalam konteks forensik dokumen, pengenalan huruf dalam dokumen rusak, atau verifikasi dokumen resmi yang memiliki watermark atau logo tertimpa teks. Teknik LBP sendiri sering digunakan dalam pengenalan wajah, deteksi tekstur industri, dan karakterisasi permukaan dalam pengolahan citra medis.

Dengan kombinasi antara antarmuka yang ramah pengguna, metode deteksi yang tangguh, serta kemampuan interaktif seperti zoom dan crop, aplikasi ini menjadi alat bantu yang kuat untuk keperluan analisis visual berbasis teks dan logo.

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import cv2
import numpy as np

class HurufLogoAnalyzer:
    def __init__(self, root):
        self.root = root
        self.root.title("Hitung Persentase Huruf Tertimpa Logo (LBP)")

        self.zoom_level = 1.0

        self.canvas_frame = tk.Frame(root)
        self.canvas_frame.pack(fill=tk.BOTH, expand=True)

        self.canvas = tk.Canvas(self.canvas_frame, bg="white",
                                cursor="cross")
        self.canvas.grid(row=0, column=0, sticky="nsew")

        self.v_scroll = tk.Scrollbar(self.canvas_frame,
                                orient=tk.VERTICAL, command=self.canvas.yview)
        self.h_scroll = tk.Scrollbar(self.canvas_frame,
                                orient=tk.HORIZONTAL, command=self.canvas.xview)
        self.v_scroll.grid(row=0, column=1, sticky="ns")
        self.h_scroll.grid(row=1, column=0, sticky="ew")

        self.canvas.config(yscrollcommand=self.v_scroll.set,
                           xscrollcommand=self.h_scroll.set)
        self.canvas_frame.grid_rowconfigure(0, weight=1)
        self.canvas_frame.grid_columnconfigure(0, weight=1)
```

```

        self.btn_frame = tk.Frame(root)
        self.btn_frame.pack(pady=5)

        self.select_btn = tk.Button(self.btn_frame, text="Pilih Gambar",
command=self.select_image)
        self.select_btn.pack(side=tk.LEFT, padx=5)

        self.result_label = tk.Label(self.btn_frame, text="Persentase
tertempa: -")
        self.result_label.pack(side=tk.LEFT, padx=10)

        self.new_instance_btn = tk.Button(self.btn_frame, text="New
Instance", command=self.create_new_instance)
        self.new_instance_btn.pack(side=tk.LEFT, padx=5)

        self.image_path = None
        self.cv_image = None
        self.tk_image = None
        self.start_x = self.start_y = self.end_x = self.end_y = 0
        self.rect = None
        self.displayed_image = None

        self.canvas.bind("<ButtonPress-1>", self.start_crop)
        self.canvas.bind("<B1-Motion>", self.update_crop)
        self.canvas.bind("<ButtonRelease-1>", self.finish_crop)
        self.canvas.bind("<MouseWheel>", self.zoom)

    def select_image(self):
        file_path = filedialog.askopenfilename(
            title="Pilih Gambar",
            filetypes=[("Image files", "*.jpg *.jpeg *.png")]
        )
        if not file_path:
            return
        self.image_path = file_path
        self.zoom_level = 1.0
        self.load_image()

    def load_image(self):
        self.cv_image = cv2.imread(self.image_path)
        self.display_image()

    def display_image(self):
        resized = cv2.resize(self.cv_image, None, fx=self.zoom_level,
fy=self.zoom_level, interpolation=cv2.INTER_LINEAR)
        image = cv2.cvtColor(resized, cv2.COLOR_BGR2RGB)
        pil_image = Image.fromarray(image)
        self.tk_image = ImageTk.PhotoImage(pil_image)

        self.canvas.delete("all")
        self.displayed_image = self.canvas.create_image(0, 0,
anchor=tk.NW, image=self.tk_image)
        self.canvas.config(scrollregion=self.canvas.bbox(tk.ALL))

    def start_crop(self, event):
        if not self.image_path:

```

```

        return
    self.start_x = self.canvas.canvasx(event.x)
    self.start_y = self.canvas.canvasy(event.y)
    if self.rect:
        self.canvas.delete(self.rect)
    self.rect = self.canvas.create_rectangle(
        self.start_x, self.start_y, self.start_x, self.start_y,
        outline="red", width=3
    )

def update_crop(self, event):
    if not self.image_path:
        return
    self.end_x = self.canvas.canvasx(event.x)
    self.end_y = self.canvas.canvasy(event.y)
    self.canvas.coords(self.rect, self.start_x, self.start_y,
self.end_x, self.end_y)

def finish_crop(self, event):
    if not self.image_path:
        return
    self.end_x = self.canvas.canvasx(event.x)
    self.end_y = self.canvas.canvasy(event.y)
    self.crop_and_analyze()

def crop_and_analyze(self):
    scale = self.zoom_level
    x1 = int(min(self.start_x, self.end_x) / scale)
    y1 = int(min(self.start_y, self.end_y) / scale)
    x2 = int(max(self.start_x, self.end_x) / scale)
    y2 = int(max(self.start_y, self.end_y) / scale)

    cropped = self.cv_image[y1:y2, x1:x2]
    if cropped.size == 0:
        messagebox.showerror("Error", "Area crop tidak valid.")
        return

    # Convert ke grayscale
    gray = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)

    # Gunakan Local Binary Pattern
    lbp = self.local_binary_pattern(gray)

    # Binarisasi hasil LBP dengan threshold
    _, thresh_huruf = cv2.threshold(lbp, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)

    # Logo menggunakan ruang warna LAB
    lab = cv2.cvtColor(cropped, cv2.COLOR_BGR2LAB)
    lower_logo = np.array([120, 130, 135])
    upper_logo = np.array([200, 170, 180])
    mask_logo = cv2.inRange(lab, lower_logo, upper_logo)

    # Morfologi untuk memperbaiki area logo
    kernel = np.ones((3, 3), np.uint8)
    mask_logo = cv2.morphologyEx(mask_logo, cv2.MORPH_CLOSE, kernel,
iterations=2)

```

```

total_huruf_pixels = cv2.countNonZero(thresh_huruf)
if total_huruf_pixels == 0:
    self.result_label.config(text="Huruf tidak terdeteksi.")
    return

overlap = cv2.bitwise_and(thresh_huruf, mask_logo)
overlap_pixels = cv2.countNonZero(overlap)

persentase = (overlap_pixels / total_huruf_pixels) * 100
self.result_label.config(
    text=f"Persentase OVERLAPPING: {persentase:.2f}%",
    font=("Arial", 30, "bold")
)

self.show_result_window(cropped, persentase)

def local_binary_pattern(self, image):
    """Menghitung LBP sederhana untuk gambar grayscale."""
    lbp_image = np.zeros_like(image)
    for y in range(1, image.shape[0] - 1):
        for x in range(1, image.shape[1] - 1):
            center = image[y, x]
            binary = (
                (image[y-1, x-1] >= center) << 7 |
                (image[y-1, x] >= center) << 6 |
                (image[y-1, x+1] >= center) << 5 |
                (image[y, x+1] >= center) << 4 |
                (image[y+1, x+1] >= center) << 3 |
                (image[y+1, x] >= center) << 2 |
                (image[y+1, x-1] >= center) << 1 |
                (image[y, x-1] >= center) << 0
            )
            lbp_image[y, x] = binary
    return lbp_image

def show_result_window(self, cropped_img, percentage):
    result_win = tk.Toplevel(self.root)
    result_win.title("Hasil Crop dan Persentase")

    h, w = cropped_img.shape[:2]
    new_height = 480
    aspect_ratio = w / h
    new_width = int(aspect_ratio * new_height)

    resized_crop = cv2.resize(cropped_img, (new_width, new_height),
    interpolation=cv2.INTER_LINEAR)
    cropped_rgb = cv2.cvtColor(resized_crop, cv2.COLOR_BGR2RGB)
    pil_cropped = Image.fromarray(cropped_rgb)
    tk_cropped = ImageTk.PhotoImage(pil_cropped)

    label_img = tk.Label(result_win, image=tk_cropped)
    label_img.image = tk_cropped
    label_img.pack(padx=10, pady=10)

    label_percent = tk.Label(
        result_win,

```

```

        text=f"Persentase OVERLAPPING: {percentage:.2f}%",
        font=("Arial", 20, "bold"),
        fg="blue"
    )
    label_percent.pack(pady=5)

    def zoom(self, event):
        if self.cv_image is not None:
            if event.delta > 0:
                self.zoom_level *= 1.1
            elif event.delta < 0:
                self.zoom_level /= 1.1
            self.zoom_level = max(0.1, min(self.zoom_level, 10))
            self.display_image()

    def create_new_instance(self):
        new_window = tk.Toplevel(self.root)
        HurufLogoAnalyzer(new_window)

if __name__ == "__main__":
    root = tk.Tk()
    app = HurufLogoAnalyzer(root)
    root.state('zoomed')
    root.mainloop()

```

HASIL IMPLEMENTASI OVERLAPPING DETECTION DENGAN LBP (Local Binary Pattern)

Berikut adalah analisis ulang hasil overlapping detection berbasis LBP (Local Binary Pattern). Analisis ini mengkaji konsistensi tekstur huruf terhadap logo, dengan fokus pada gangguan visual (overlap) yang dapat mengindikasikan keaslian atau manipulasi.

Dari hasil deteksi pada huruf-huruf di ijazah kiri (Frono Jiwo), beberapa huruf seperti "A", "O", "I", "D", dan "E" menunjukkan nilai overlapping yang cukup tinggi, seperti 15.04%, 10.20%, 47.64%, 25.51%, dan 27.02%. Meskipun angka-angka ini tergolong besar, ini bisa dijelaskan oleh proses pencetakan dokumen orisinal yang memang mencetak logo di atas teks, sebagai bentuk sistem pengamanan legal pada dokumen akademik.

Fenomena di mana teks tertimpa sebagian oleh segel bukanlah hal yang aneh dalam dokumen sah. Banyak institusi menerapkan prosedur ini untuk mencegah pemalsuan, di mana segel fisik atau cetak transparan

sebagian menindih teks sebagai bukti bahwa dokumen belum dimodifikasi.



Sebaliknya, ijazah milik Joko Widodo (kanan) menunjukkan huruf-huruf dengan nilai overlapping yang sangat rendah seperti 1.90%, 2.94%, 0.13%, hingga 0.00%. Pada pandangan awal, ini mungkin terkesan ideal karena teks tidak terganggu. Namun dalam konteks forensik dokumen, justru ketiadaan overlap bisa menjadi indikasi manipulasi atau rekonstruksi digital.

Dalam ijazah sah, segel atau logo biasanya tidak dihindari dari teks, tetapi justru disusun agar menindih bagian penting secara halus. Hal ini penting

untuk menghindari pemalsuan dokumen, karena memisahkan logo dari teks sepenuhnya justru memudahkan duplikasi dan rekayasa konten.

Hasil overlapping yang terlalu "bersih" dalam ijazah Joko Widodo memperlihatkan bahwa kemungkinan logo ditempatkan secara digital setelah teks selesai diketik, bukan bagian dari cetakan resmi awal. Dengan tidak adanya overlap signifikan, dokumen ini kehilangan karakteristik penting dari sistem pengamanan institusional.

Bahkan pada huruf seperti "E", hasil deteksi menunjukkan nilai 0.00%, yang bisa berarti bahwa logo benar-benar tidak menyentuh huruf mana pun. Ini adalah anomali dalam dokumen resmi, apalagi jika segel berada di tengah dan teks berada di jalurnya. Dalam ijazah sah, minimal akan ada kontak kecil.

Huruf "A", "E", dan "I" di ijazah kanan juga memperlihatkan nilai overlapping di bawah 3%, dan ini tidak menunjukkan gangguan struktural sama sekali. Dengan tidak adanya tumpang tindih, sangat mungkin bahwa dokumen disusun ulang secara grafis, atau dicetak ulang setelah menghapus atau mengatur ulang posisi segel.

Sebaliknya, ijazah kiri meskipun tampak terganggu, justru menunjukkan bahwa teks dan segel dicetak bersamaan dalam satu proses, tanpa penghindaran satu sama lain. Hal ini umum ditemukan pada ijazah asli, sebagai tanda bahwa seluruh elemen dokumen merupakan satu kesatuan sah.

Huruf "I" dengan overlapping 47.64% pada ijazah Frono Jiwo sangat mungkin berada tepat di atas lambang universitas, memperlihatkan bahwa desain asli memang menyisipkan teks melalui area segel. Ini justru memperkuat keaslian, bukan melemahkannya.

Hal yang sama berlaku pada huruf "D", "E", dan "A" lainnya yang memiliki overlapping tinggi. Alih-alih dianggap sebagai cacat desain, hal ini menandakan bahwa tidak ada upaya untuk memanipulasi posisi teks agar

menghindari segel. Tata letak seperti ini sulit dihasilkan jika bukan dari sumber resmi.

LBP digunakan untuk mendeteksi tekstur dari huruf yang terganggu. Ketika huruf tertimpa oleh bagian logo, maka pola LBP-nya akan berubah drastis dan terdeteksi sebagai nilai abnormal. Inilah mengapa overlapping bisa dihitung dan dijadikan bukti visual dan numerik.

Jika dalam ijazah Joko Widodo nilai LBP konsisten di seluruh huruf, bahkan pada area dekat logo, ini menjadi pertanda bahwa tekstur tidak terganggu, atau dengan kata lain logo disisipkan belakangan secara digital, bukan bagian dari pencetakan dokumen awal.

Dokumen resmi yang sah biasanya memiliki banyak inkonsistensi mikro (noise, cetakan sedikit buram, logo menyentuh teks, dll) yang sulit ditiru secara digital. Ijazah Frono Jiwo menunjukkan semua karakteristik ini secara alami, termasuk tumpang tindih antar elemen.

Ijazah Joko Widodo tampak terlalu bersih dan steril, sesuatu yang sangat jarang ditemui dalam dokumen cetak zaman 1980-an. Hal ini patut dicurigai sebagai hasil rekayasa modern yang mencoba menyerupai dokumen lama, namun gagal menyertakan aspek overprinting asli.

Segel yang menindih teks merupakan salah satu elemen pengaman visual. Dalam ijazah Frono Jiwo, ini dilakukan pada banyak huruf seperti "A", "I", "E", dan "D", yang memperkuat keaslian dan menyulitkan proses pemalsuan.

Sebaliknya, dalam dokumen palsu, desainer sering kali tidak berani menindih teks dengan logo karena khawatir kehilangan keterbacaan, atau karena teknis digital yang tidak mendukung blending segel secara alami.

Hasil ini juga menunjukkan pentingnya metode LBP dalam forensik dokumen. Dengan mengamati ketidakrataan tekstur akibat gangguan

dari elemen lain, LBP dapat menunjukkan apakah suatu huruf telah mengalami overlay dari stempel asli atau tidak.

Tidak hanya itu, penggunaan ruang warna LAB untuk deteksi segel memungkinkan sistem mengenali logo meskipun warna latarnya mirip atau memiliki intensitas rendah. Kombinasi dua metode ini sangat efektif dalam mendeteksi manipulasi digital.

Dari segi forensik visual, dokumen yang memiliki nilai overlapping tinggi namun tetap terbaca, lebih sah daripada dokumen yang terlalu sempurna namun menunjukkan indikasi isolasi teks dari segel.

Ijazah Frono Jiwo juga menunjukkan konsistensi antar huruf, meskipun ada yang memiliki overlapping tinggi, namun tidak terlihat pola manipulasi. Ini berlawanan dengan ijazah Joko Widodo yang secara tekstur terlihat seragam namun artifisial.

Dengan demikian, dalam konteks deteksi keaslian dokumen menggunakan metode LBP dan segmentasi LAB, ijazah milik Frono Jiwo (kiri) dapat disimpulkan sebagai asli, sementara ijazah milik Joko Widodo (kanan) menunjukkan indikasi kuat sebagai dokumen palsu.

Kesimpulan ini diperkuat oleh fakta bahwa overlapping secara alami terjadi dalam proses cetak resmi, sedangkan ketiadaan overlapping justru menjadi tanda adanya manipulasi atau pembuatan ulang. Maka, analisis berbasis citra ini sangat mendukung asumsi bahwa dokumen kiri adalah asli, dan dokumen kanan adalah hasil rekayasa.

IMPLEMENTASI OVERLAPPING DETECTION DENGAN CANNY EDGE DETECTOR DAN DILATION

Program berikut memanfaatkan pendekatan modern dalam bidang forensik digital gambar untuk menganalisis apakah area huruf pada dokumen tertimpa oleh logo, yang bisa menjadi indikasi adanya manipulasi dokumen. Salah satu kegunaan utamanya dalam dunia

forensik adalah sebagai alat bantu identifikasi pemalsuan pada dokumen seperti ijazah, sertifikat, atau surat penting lainnya, dengan menilai keberadaan overlap antara teks dan elemen grafis (seperti logo institusi).

Metode utama yang digunakan adalah Canny Edge Detection, sebuah teknik deteksi tepi yang sangat terkenal dalam visi komputer karena ketajamannya dalam menemukan batas objek, termasuk huruf-huruf dalam citra grayscale. Dalam konteks ini, Canny digunakan untuk mengekstraksi struktur bentuk huruf, dengan hasil berupa representasi tepi huruf-huruf yang ada dalam area yang di-crop pengguna.

Untuk meningkatkan representasi huruf setelah pendeteksian tepi, program menerapkan dilasi morfologis, yaitu operasi citra yang mempertebal objek putih dalam gambar biner. Ini membantu memperkuat area huruf yang telah dikenali sebagai tepi tipis oleh Canny agar menjadi blok-blok pixel yang cukup besar untuk dihitung sebagai huruf valid.

Langkah berikutnya adalah deteksi area logo. Di sini digunakan pendekatan ruang warna LAB, yang lebih stabil terhadap pencahayaan dibanding RGB. Dengan menentukan kisaran nilai LAB yang merepresentasikan warna umum dari logo, program membuat masker biner area mana dalam gambar crop yang termasuk dalam warna khas logo, terlepas dari pencahayaan atau perbedaan latar belakang.

Masker logo tersebut kemudian diperhalus menggunakan morphological closing, yaitu gabungan dilasi dan erosi, yang akan menutupi celah kecil dalam area logo dan menghubungkan komponen logo yang terputus karena noise atau ketidaksinambungan warna. Hasilnya adalah sebuah wilayah logo yang lebih solid dan mewakili area penuh di mana logo tertimpa.

Setelah huruf dan logo terdeteksi, keduanya dibandingkan menggunakan operasi bitwise AND, yang hanya menyisakan area di mana keduanya tumpang tindih. Overlap inilah yang menjadi inti dari analisis forensik

dalam program ini, karena menunjukkan secara visual dan numerik berapa besar bagian huruf yang tertutupi oleh logo.

Dari overlap tersebut, program menghitung persentase jumlah piksel huruf yang tertimpa logo dibanding total huruf. Angka inilah yang ditampilkan secara visual ke pengguna. Semakin tinggi nilai ini, semakin besar kemungkinan logo ditambahkan secara digital setelah teks ditulis atau dipindai.

Persentase ini bisa menjadi indikator kuat dalam investigasi forensik: dalam dokumen asli, huruf biasanya dicetak setelah logo, atau keduanya dicetak bersama dalam satu cetakan. Jika huruf tampak "tersembunyi" di bawah logo digital, bisa jadi ada upaya rekayasa grafis yang disengaja.

Antarmuka pengguna dibangun dengan Tkinter, menyediakan kemudahan bagi analis untuk memuat dokumen, men-zoom, memilih area crop dengan mouse, dan menampilkan hasil analisis dalam jendela terpisah. Fitur ini penting dalam konteks forensik karena memungkinkan interaktivitas dan perbandingan visual secara langsung tanpa coding ulang.

Fungsi zoom memberikan fleksibilitas tinggi untuk memperbesar detail halus dalam dokumen, yang penting dalam mendeteksi manipulasi mikro seperti spasi ganjil, piksel tidak natural, atau artefak edge tajam yang bisa menunjukkan hasil rekayasa gambar.

Program juga memungkinkan multi-instances dari GUI dengan membuat jendela analisis baru tanpa menutup aplikasi utama, fitur berguna untuk menganalisis banyak dokumen sekaligus dan membandingkannya berdampingan dalam kasus forensik skala besar.

Keunggulan lainnya adalah metode ini tidak bergantung pada OCR atau teks eksplisit, tetapi murni berbasis visual dan struktur spasial, sehingga tetap efektif meskipun fontnya tidak standar, orientasinya miring, atau terdapat noise pemindaian.

Teknik Canny Edge ini secara umum lebih robust terhadap variasi latar belakang atau kecerahan dibanding metode lama seperti thresholding global atau LBP (Local Binary Pattern). Ini membuatnya lebih akurat dalam dokumen yang sudah tua, pudar, atau hasil pemindaian tidak sempurna.

Aplikasi ini dapat diperluas untuk menyimpan bukti visual berupa gambar hasil crop beserta area overlap yang ditandai warna tertentu, untuk digunakan dalam laporan investigasi resmi atau presentasi bukti forensik.

Selain itu, pendekatan ini dapat digabung dengan teknik Machine Learning seperti klasifikasi CNN untuk mengotomatisasi keputusan “asli atau palsu” berdasarkan pola overlap huruf dan logo dalam ribuan dokumen yang dianalisis.

Program ini juga bisa diterapkan dalam pengawasan kualitas cetak dokumen resmi di instansi pendidikan atau pemerintahan, memastikan bahwa prosedur pencetakan dokumen sesuai urutan dan tidak ditambah logo digital setelahnya.

Untuk keperluan pengadilan, kode ini dapat digunakan untuk menyediakan bukti kuantitatif bahwa bagian teks tertentu tidak konsisten dengan proses pencetakan asli, mendukung atau menantang validitas dokumen yang dipertanyakan.

Fitur crop manual sangat penting, karena memungkinkan analisis fokus hanya pada bagian dokumen yang relevan, seperti area tanda tangan, kop institusi, atau nomor ijazah, tanpa terganggu oleh elemen lain dalam citra.

Metode ini bersifat non-destruktif dan tidak mengubah citra asli. Hal ini penting dari sisi hukum karena menjaga keutuhan bukti digital sebagai prasyarat dalam keabsahan di pengadilan.

Dengan pendekatan open source dan antarmuka visual yang user-friendly, program ini memungkinkan para analis forensik, pengacara, atau bahkan wartawan investigatif untuk melakukan analisis mandiri terhadap dokumen digital yang diduga palsu.

Terakhir, penggunaan Python dan pustaka populer seperti OpenCV, PIL, dan Tkinter memungkinkan pengembangan lanjutan yang cepat, seperti menambahkan anotasi, laporan PDF otomatis, atau pencocokan logo database untuk deteksi logo palsu.

Secara keseluruhan, program ini mencerminkan pendekatan modern dan efisien dalam forensic document examination berbasis citra digital, khususnya dalam mendeteksi manipulasi logo terhadap teks dalam dokumen resmi. Ini menjadikannya alat yang berpotensi penting dalam investigasi integritas dokumen publik dan hukum.

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import cv2
import numpy as np

class HurufLogoAnalyzer:
    def __init__(self, root):
        self.root = root
        self.root.title("Hitung Persentase Huruf Tertimpa Logo (Canny + Dilation)")

        self.zoom_level = 1.0

        self.canvas_frame = tk.Frame(root)
        self.canvas_frame.pack(fill=tk.BOTH, expand=True)

        self.canvas = tk.Canvas(self.canvas_frame, bg="white",
                                cursor="cross")
        self.canvas.grid(row=0, column=0, sticky="nsew")

        self.v_scroll = tk.Scrollbar(self.canvas_frame,
                                     orient=tk.VERTICAL, command=self.canvas.yview)
        self.h_scroll = tk.Scrollbar(self.canvas_frame,
                                     orient=tk.HORIZONTAL, command=self.canvas.xview)
        self.v_scroll.grid(row=0, column=1, sticky="ns")
        self.h_scroll.grid(row=1, column=0, sticky="ew")

        self.canvas.config(yscrollcommand=self.v_scroll.set,
                           xscrollcommand=self.h_scroll.set)
        self.canvas_frame.grid_rowconfigure(0, weight=1)
        self.canvas_frame.grid_columnconfigure(0, weight=1)

        self.btn_frame = tk.Frame(root)
        self.btn_frame.pack(pady=5)

        self.select_btn = tk.Button(self.btn_frame, text="Pilih Gambar",
                                     command=self.select_image)
        self.select_btn.pack(side=tk.LEFT, padx=5)
```



```

        self.result_label = tk.Label(self.btn_frame, text="Persentase
tertinggal: -")
        self.result_label.pack(side=tk.LEFT, padx=10)

        self.new_instance_btn = tk.Button(self.btn_frame, text="New
Instance", command=self.create_new_instance)
        self.new_instance_btn.pack(side=tk.LEFT, padx=5)

        self.image_path = None
        self.cv_image = None
        self.tk_image = None
        self.start_x = self.start_y = self.end_x = self.end_y = 0
        self.rect = None
        self.displayed_image = None

        self.canvas.bind("<ButtonPress-1>", self.start_crop)
        self.canvas.bind("<B1-Motion>", self.update_crop)
        self.canvas.bind("<ButtonRelease-1>", self.finish_crop)
        self.canvas.bind("<MouseWheel>", self.zoom)

    def select_image(self):
        file_path = filedialog.askopenfilename(
            title="Pilih Gambar",
            filetypes=[("Image files", "*.jpg *.jpeg *.png")]
        )
        if not file_path:
            return
        self.image_path = file_path
        self.zoom_level = 1.0
        self.load_image()

    def load_image(self):
        self.cv_image = cv2.imread(self.image_path)
        self.display_image()

    def display_image(self):
        resized = cv2.resize(self.cv_image, None, fx=self.zoom_level,
fy=self.zoom_level, interpolation=cv2.INTER_LINEAR)
        image = cv2.cvtColor(resized, cv2.COLOR_BGR2RGB)
        pil_image = Image.fromarray(image)
        self.tk_image = ImageTk.PhotoImage(pil_image)

        self.canvas.delete("all")
        self.displayed_image = self.canvas.create_image(0, 0,
anchor=tk.NW, image=self.tk_image)
        self.canvas.config(scrollregion=self.canvas.bbox(tk.ALL))

    def start_crop(self, event):
        if not self.image_path:
            return
        self.start_x = self.canvas.canvasx(event.x)
        self.start_y = self.canvas.canvasy(event.y)
        if self.rect:
            self.canvas.delete(self.rect)
        self.rect = self.canvas.create_rectangle(
            self.start_x, self.start_y, self.start_x, self.start_y,

```

```

        outline="red", width=3
    )

    def update_crop(self, event):
        if not self.image_path:
            return
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.canvas.coords(self.rect, self.start_x, self.start_y,
self.end_x, self.end_y)

    def finish_crop(self, event):
        if not self.image_path:
            return
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.crop_and_analyze()

    def crop_and_analyze(self):
        scale = self.zoom_level
        x1 = int(min(self.start_x, self.end_x) / scale)
        y1 = int(min(self.start_y, self.end_y) / scale)
        x2 = int(max(self.start_x, self.end_x) / scale)
        y2 = int(max(self.start_y, self.end_y) / scale)

        cropped = self.cv_image[y1:y2, x1:x2]
        if cropped.size == 0:
            messagebox.showerror("Error", "Area crop tidak valid.")
            return

        gray = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)

        # 🔍 Gunakan metode modern: Canny Edge Detection + Dilation
        edges = cv2.Canny(gray, 50, 150)
        kernel = np.ones((2, 2), np.uint8)
        thresh_huruf = cv2.dilate(edges, kernel, iterations=1)

        # 🎯 Deteksi area logo dalam ruang warna LAB
        lab = cv2.cvtColor(cropped, cv2.COLOR_BGR2LAB)
        lower_logo = np.array([120, 130, 135])
        upper_logo = np.array([200, 170, 180])
        mask_logo = cv2.inRange(lab, lower_logo, upper_logo)

        # 🔧 Morfologi untuk memperbaiki area logo
        kernel_logo = np.ones((3, 3), np.uint8)
        mask_logo = cv2.morphologyEx(mask_logo, cv2.MORPH_CLOSE,
kernel_logo, iterations=2)

        total_huruf_pixels = cv2.countNonZero(thresh_huruf)
        if total_huruf_pixels == 0:
            self.result_label.config(text="Huruf tidak terdeteksi.")
            return

        overlap = cv2.bitwise_and(thresh_huruf, mask_logo)
        overlap_pixels = cv2.countNonZero(overlap)

```

```

    persentase = (overlap_pixels / total_huruf_pixels) * 100
    self.result_label.config(
        text=f"Persentase OVERLAPPING: {persentase:.2f}%",
        font=("Arial", 30, "bold")
    )

    self.show_result_window(cropped, persentase)

def show_result_window(self, cropped_img, percentage):
    result_win = tk.Toplevel(self.root)
    result_win.title("Hasil Crop dan Persentase")

    h, w = cropped_img.shape[:2]
    new_height = 480
    aspect_ratio = w / h
    new_width = int(aspect_ratio * new_height)

    resized_crop = cv2.resize(cropped_img, (new_width, new_height),
    interpolation=cv2.INTER_LINEAR)
    cropped_rgb = cv2.cvtColor(resized_crop, cv2.COLOR_BGR2RGB)
    pil_cropped = Image.fromarray(cropped_rgb)
    tk_cropped = ImageTk.PhotoImage(pil_cropped)

    label_img = tk.Label(result_win, image=tk_cropped)
    label_img.image = tk_cropped
    label_img.pack(padx=10, pady=10)

    label_percent = tk.Label(
        result_win,
        text=f"Persentase OVERLAPPING: {percentage:.2f}%",
        font=("Arial", 20, "bold"),
        fg="blue"
    )
    label_percent.pack(pady=5)

def zoom(self, event):
    if self.cv_image is not None:
        if event.delta > 0:
            self.zoom_level *= 1.1
        elif event.delta < 0:
            self.zoom_level /= 1.1
        self.zoom_level = max(0.1, min(self.zoom_level, 10))
        self.display_image()

def create_new_instance(self):
    new_window = tk.Toplevel(self.root)
    HurufLogoAnalyzer(new_window)

if __name__ == "__main__":
    root = tk.Tk()
    app = HurufLogoAnalyzer(root)
    root.state('zoomed')
    root.mainloop()

```

HASIL IMPLEMENTASI OVERLAPPING DETECTION DENGAN CANNY EDGE DETECTOR DAN DILATION

Gambar yang ditampilkan memperlihatkan dua dokumen ijazah dari Universitas Gadjah Mada, masing-masing atas nama Frono Jiwo (kiri) dan Joko Widodo (kanan). Untuk menganalisis keaslian dokumen tersebut secara forensik, digunakan metode overlapping detection dengan kombinasi Canny edge detection dan morfologi dilasi. Teknik ini sangat efektif untuk mengukur sejauh mana huruf dalam dokumen tertimpa oleh logo, yang berpotensi mengindikasikan manipulasi digital.



Prosedur analisis ini dimulai dengan cropping area huruf yang berada di atas logo. Kemudian, gambar diubah ke dalam format grayscale untuk

menyederhanakan analisis spasial piksel. Dengan metode Canny edge, tepi huruf dapat dikenali dengan presisi tinggi, tanpa dipengaruhi noise pencahayaan yang berlebih.

Setelah tepi huruf dikenali, proses dilasi diterapkan untuk memperbesar area tepi sehingga menghasilkan representasi huruf yang lebih utuh. Ini penting agar jumlah piksel huruf bisa dihitung secara representatif saat dibandingkan dengan area logo.

Sementara itu, untuk mendeteksi area logo, gambar diubah ke ruang warna LAB. Ini memberikan kelebihan dalam stabilitas pencahayaan dan lebih presisi dalam membedakan warna emas atau coklat tua khas logo ijazah. Warna-warna khas logo kemudian dijadikan range mask dalam proses segmentasi.

Masker logo yang dihasilkan kemudian disempurnakan dengan operasi morfologi closing. Tujuannya adalah untuk menutupi lubang kecil dan menyatukan bagian logo yang terpisah, sehingga diperoleh area logo yang utuh dan solid untuk dihitung dalam analisis overlap.

Dua mask—satu dari huruf dan satu dari logo—kemudian dibandingkan menggunakan operasi bitwise AND. Hasil dari operasi ini memberikan citra area yang merupakan irisan antara huruf dan logo, alias wilayah tertimpa.

Analisis kemudian mengukur berapa banyak piksel dalam huruf yang tertimpa logo dibandingkan dengan total piksel huruf, lalu menampilkan persentase overlap sebagai hasil numerik. Semakin tinggi nilai ini, semakin besar kemungkinan huruf tertimpa oleh logo secara tidak wajar.

Pada hasil untuk ijazah Frono Jiwo, dapat diamati bahwa beberapa huruf seperti “I”, “E”, “A”, dan “D” memiliki nilai overlap yang sangat tinggi, berkisar antara 27% hingga 56%. Ini menunjukkan bahwa logo memang dicetak setelah huruf atau dalam proses yang bersamaan, seperti yang wajar terjadi dalam pencetakan dokumen asli.

Contohnya, huruf "I" pada ijazah Frono menunjukkan overlap sebesar 56.27%, angka yang sangat tinggi. Ini berarti sebagian besar bagian huruf benar-benar tertimpa oleh area logo, mendukung dugaan bahwa logo dan teks dicetak bersama dalam satu proses.

Huruf "E" dari ijazah Frono juga menunjukkan 32.86% overlap, menguatkan kesan autentik karena pencetakan logo telah menimpa bagian tubuh huruf dengan konsisten dan nyata. Nilai yang tinggi ini sulit dicapai oleh manipulasi digital kecuali dengan teknik pencetakan ulang profesional.

Sebaliknya, pada ijazah Joko Widodo, persentase overlap huruf terhadap logo umumnya sangat rendah. Contohnya huruf "O" hanya menunjukkan overlap sebesar 1.00%, "A" sebesar 7.69%, dan huruf "E" hanya 4.15%. Angka ini secara statistik tergolong rendah dan mencurigakan dalam konteks pencetakan dokumen resmi.

Hal ini memperlihatkan bahwa logo tampak seperti berada di bawah huruf, bukan di atasnya. Ini berlawanan dengan alur pencetakan dokumen resmi yang pada umumnya mencetak logo terlebih dahulu atau secara bersamaan dengan teks.

Huruf "D" pada ijazah Joko juga hanya menunjukkan overlap sebesar 8.97%, dan huruf "I" sebesar 10.80%. Walaupun sedikit lebih tinggi dibanding huruf lainnya, nilai ini masih jauh di bawah angka overlap pada dokumen Frono Jiwo.

Ketimpangan dalam persentase overlap ini menjadi indikator penting bahwa huruf-huruf pada ijazah Joko Widodo mungkin telah ditambahkan atau direkonstruksi setelah logo dicetak, atau sebaliknya, logonya ditempel kemudian secara digital untuk menyembunyikan kekosongan atau manipulasi lainnya.

Dalam kasus ijazah asli, kita mengharapkan adanya overlap alami dan signifikan antara logo dan huruf karena printer biasanya mencetak dalam satu kali proses yang melibatkan semua elemen dokumen. Nilai-nilai di

atas 20% seperti pada ijazah Frono menandakan konsistensi dengan proses cetak dokumen resmi.

Perbedaan mencolok antara kedua dokumen ini tidak hanya secara visual tetapi juga secara kuantitatif, menunjukkan bahwa struktur dokumen pada ijazah Joko Widodo kemungkinan besar dihasilkan dari proses manipulasi grafis.

Selain itu, tidak terdapat pola alami ink bleeding atau noise khas hasil cetak pada huruf yang tertimpa logo dalam ijazah Joko Widodo, sementara hal itu dapat diamati pada area overlap ijazah Frono. Ini memperkuat dugaan penggunaan proses digital dalam dokumen yang kanan.

Perlu dicatat bahwa penggunaan edge detector seperti Canny memiliki keunggulan dalam mendeteksi tepi objek secara tajam meskipun dalam kondisi pencahayaan bervariasi. Ini memastikan bahwa struktur huruf terdeteksi meski pencetakan kurang sempurna atau hasil scan tidak terlalu jelas.

Proses dilasi juga memegang peran penting karena memastikan bahwa piksel-piksel tepi huruf yang semula hanya beberapa baris dapat diubah menjadi blok pixel yang lebih lebar, sehingga hasil perbandingan dengan area logo menjadi valid.

Teknik ini memiliki keunggulan dibanding LBP karena tidak hanya mendeteksi tekstur lokal melainkan mengandalkan representasi spasial langsung dari bentuk huruf dan logo. Dalam konteks analisis dokumen resmi, ini membuatnya lebih akurat dan relevan.

Citra hasil crop yang diperbesar dan diberi label persentase overlap memberikan dukungan visual yang kuat dalam pelaporan forensik. Hal ini penting karena memungkinkan auditor, pengacara, atau pengamat publik untuk memahami hasil analisis tanpa latar belakang teknis.

Dari sisi hukum dan akademik, pendekatan ini memberikan bukti kuantitatif objektif yang bisa diperiksa dan diulang. Bukti ini memiliki potensi untuk menjadi bagian dari audit forensik atau investigasi resmi terhadap keaslian dokumen publik.

Konsistensi tinggi dari overlap pada ijazah Frono Jiwo memperlihatkan bahwa dokumen tersebut dicetak secara sah, menggunakan mesin dan prosedur resmi yang memastikan bahwa semua elemen saling tumpang tindih sesuai urutan pencetakan.

Sebaliknya, inkonsistensi nilai overlap dan posisi grafis pada ijazah Joko Widodo mengarah pada kesimpulan sementara bahwa dokumen ini kemungkinan besar merupakan hasil manipulasi atau rekonstruksi digital.

Program analisis ini bekerja non-destruktif, artinya tidak mengubah data asli, hanya mengambil analisis dari area yang dipilih pengguna. Ini penting dari sisi metodologi forensik karena menjaga integritas bukti asli. Akhirnya, dengan menggabungkan visualisasi, kuantifikasi, dan teknologi edge detection, hasil analisis ini memberikan **dukungan kuat bahwa dokumen sebelah kiri (Frono Jiwo) adalah asli, sementara dokumen sebelah kanan (Joko Widodo) sangat mungkin palsu.**

ANALISIS LAPISAN VISUAL

TENTANG ANALISIS LAPISAN VISUAL

Analisis lapisan visual merupakan salah satu teknik penting dalam forensik digital dokumen, yang digunakan untuk menentukan urutan atau posisi elemen visual dalam sebuah gambar dokumen—seperti teks, logo, stempel, tanda tangan, atau latar belakang. Tujuan utama dari analisis ini adalah untuk mengetahui apakah suatu elemen dicetak atau ditambahkan terlebih dahulu, atau apakah ia ditindih oleh elemen lain. Informasi ini sangat krusial untuk mengidentifikasi apakah suatu dokumen telah dimanipulasi atau dipalsukan.

Dalam konteks ijazah, sertifikat, surat resmi, atau dokumen berstempel lainnya, analisis lapisan visual digunakan untuk mengevaluasi apakah tulisan atau huruf dicetak setelah atau sebelum logo atau stempel. Ini penting karena stempel resmi biasanya ditambahkan terakhir setelah isi dokumen selesai dicetak. Jika sebuah huruf tampak berada di atas stempel padahal secara logika seharusnya tidak, hal tersebut bisa menjadi indikasi manipulasi.

Salah satu metode paling dasar dalam analisis lapisan visual adalah inspeksi visual oleh ahli forensik dokumen dengan bantuan kaca pembesar atau mikroskop digital. Namun, metode ini bersifat subjektif dan sangat bergantung pada keahlian si pengamat. Oleh karena itu, metode berbasis citra digital kini semakin diandalkan karena memberikan hasil kuantitatif dan dapat direproduksi secara objektif.

Metode populer dalam analisis digital adalah Color Clustering, seperti K-Means Clustering. Teknik ini memisahkan gambar dokumen berdasarkan dominasi warna dalam klaster tertentu. Dengan memisahkan klaster warna, kita bisa mengidentifikasi kelompok piksel yang seragam—misalnya, huruf hitam dengan logo berwarna cokelat kemerahan. Jika suatu area tampak mengandung campuran warna dari dua klaster, maka area tersebut bisa jadi merupakan titik overlapping atau tumpang tindih antara dua objek visual.

Metode K-Means Clustering bekerja dengan mengelompokkan piksel berdasarkan kemiripan warna (biasanya pada ruang warna RGB atau LAB). Setelah dikelompokkan, hasilnya divisualisasikan dalam bentuk representasi terpisah. Klaster teks akan tampak menonjol sendiri, dan klaster logo pun akan terlihat sebagai entitas terpisah. Interseksi antara dua klaster ini bisa menunjukkan area yang tertimpa atau menimpa.

Metode lain yang sering digunakan dalam analisis lapisan visual adalah Canny Edge Detection dikombinasikan dengan operasi Dilation. Canny digunakan untuk mendeteksi tepi-tepi tajam pada huruf atau logo, sementara Dilation melebarkan area tersebut sehingga menghasilkan representasi visual yang tegas dari objek. Ini membantu mengukur tumpang tindih antar elemen berdasarkan keberadaan piksel edge yang bersinggungan.

Selain itu, teknik Layer Separation juga digunakan dalam forensik dokumen digital. Layer separation biasanya dilakukan dengan pendekatan filter spasial atau analisis frekuensi, yang memungkinkan pemisahan komponen berdasarkan intensitas pencetakan, tekstur, dan noise pola tinta. Misalnya, mesin cetak menghasilkan tekstur tertentu yang berbeda dengan tinta stempel manual.

Metode Multispectral Imaging juga sering diterapkan dalam laboratorium forensik. Teknik ini menggunakan pencahayaan dengan berbagai panjang gelombang—from ultraviolet, cahaya tampak, hingga inframerah—untuk mengungkap lapisan tinta yang tersembunyi. Tinta yang berbeda akan

memiliki respons spektral yang berbeda, sehingga bisa diidentifikasi urutan penggunaannya.

Selanjutnya, Hyperspectral Imaging adalah versi lanjutan dari multispektral. Dengan menangkap ratusan saluran spektral, metode ini bisa membedakan tinta dengan sensitivitas tinggi, bahkan jika perbedaan warna tidak terlihat oleh mata manusia. Teknik ini sangat mahal dan kompleks, tapi sangat andal dalam menentukan urutan penulisan dalam dokumen.

Metode X-ray Fluorescence (XRF) juga digunakan untuk mengetahui komposisi kimia tinta dan bahan dasar dokumen. XRF dapat membedakan tinta berdasarkan kandungan logam dan unsur kimia yang digunakan dalam formulanya. Jika dua area tumpang tindih memiliki komposisi yang sama atau berbeda, maka dapat diketahui mana yang dicetak terlebih dahulu.

Dalam ranah digital, Histogram Analysis juga dapat digunakan untuk membandingkan intensitas warna antara area tumpang tindih dan area normal. Perbedaan distribusi intensitas bisa menunjukkan adanya lapisan yang menimpa satu sama lain.

Teknik Optical Character Recognition (OCR) + Masking juga digunakan untuk mengenali huruf, kemudian diterapkan masking pada area yang diduga tertimpa. Area tersebut dibandingkan dengan latar belakang atau lapisan di bawahnya. Ini membantu mengidentifikasi jika ada area yang seharusnya kosong namun tertimpa huruf.

Ada pula metode berbasis Machine Learning yang menggunakan klasifikasi supervised untuk membedakan bagian dokumen yang otentik dan yang telah dimanipulasi. Dengan melatih model pada dataset dokumen asli dan palsu, model dapat belajar mengenali pola tumpang tindih atau manipulasi lapisan secara otomatis.

Metode Principal Component Analysis (PCA) dapat diterapkan untuk mengurangi dimensi warna dan menemukan pola dominan dalam citra

dokumen. PCA dapat membantu dalam visualisasi area yang memiliki anomali komponen warna, yang menunjukkan perbedaan lapisan cetakan.

Wavelet Transform adalah metode lain yang memungkinkan analisis multi-resolusi. Dengan memecah gambar ke dalam beberapa skala, kita dapat mendeteksi ketidaksesuaian tekstur atau noise yang disebabkan oleh tumpang tindih atau manipulasi.

Selain pendekatan warna dan tepi, analisis Noise Residual digunakan dalam deteksi manipulasi. Setiap printer atau scanner meninggalkan noise spesifik pada hasil cetak atau hasil scan. Jika satu bagian dokumen memiliki noise yang berbeda, maka bisa diindikasikan bahwa bagian itu ditambahkan kemudian.

Dalam beberapa kasus, teknik Photometric Stereo diterapkan untuk menciptakan model kedalaman dari permukaan dokumen. Teknik ini digunakan untuk menganalisis apakah permukaan tinta menonjol atau rata, yang bisa mengindikasikan lapisan mana yang ditulis lebih dahulu. Untuk meningkatkan akurasi, sering kali digunakan gabungan dari beberapa teknik di atas. Misalnya, Color Clustering dikombinasikan dengan Edge Detection dan Histogram Matching untuk mendapatkan hasil yang lebih menyeluruh. Ini karena pemalsuan dokumen seringkali sangat halus dan memerlukan pendekatan multidimensi.

Penerapan GUI (Graphical User Interface) seperti dalam program Anda membantu mempercepat dan mempermudah analisis visual lapisan dokumen. Pengguna bisa langsung melihat hasil clustering, tepi, dan overlapping dalam satu tampilan, serta mendapatkan kesimpulan berbasis persentase tumpang tindih antara huruf dan logo.

Terakhir, penting untuk dicatat bahwa hasil dari analisis lapisan visual sebaiknya didukung dengan data metadata dan uji laboratorium lainnya, terutama jika hasil analisis digunakan dalam proses hukum. Analisis ini merupakan bagian penting dari proses verifikasi otentisitas dokumen, namun bukan satu-satunya.

Dengan memadukan teknik digital modern dan pendekatan forensik klasik, analisis lapisan visual menjadi alat yang sangat efektif untuk mengungkap pemalsuan dokumen yang dilakukan secara digital maupun analog. Pendekatan ini terus berkembang seiring kemajuan teknologi pengolahan citra dan kecerdasan buatan.

ANALISIS LAPISAN VISUAL DENGAN K-MEANS CLUSTERING

Program forensik visual menggunakan metode K-Means Clustering dalam kode di atas dirancang untuk menganalisis tumpang tindih antara huruf dan logo pada dokumen dengan memanfaatkan perbedaan warna. Tujuan utama dari metode ini adalah untuk membantu menentukan urutan pencetakan elemen-elemen dalam dokumen digital berdasarkan distribusi warna. Kode ini dapat mengungkap apakah teks dicetak di atas logo atau sebaliknya, yang merupakan informasi penting dalam investigasi forensik dokumen.

Secara teknis, analisis dilakukan dengan cara memotong bagian dari gambar yang dipilih oleh pengguna menggunakan mouse. Setelah area dipilih, data warna piksel dalam area tersebut dikumpulkan dan diolah menggunakan algoritma K-Means. Metode ini bekerja dengan mengelompokkan piksel berdasarkan kesamaan warna dalam ruang RGB (Red, Green, Blue). Warna-warna yang mirip akan tergabung dalam satu klaster.

Clustering dilakukan dengan parameter $n_clusters=3$, yang berarti citra akan dikelompokkan menjadi tiga kelompok warna dominan. Tiga warna ini umumnya mewakili latar belakang, teks (huruf), dan logo. Ini adalah jumlah ideal untuk kasus di mana kita hanya ingin membedakan tiga lapisan visual utama dalam dokumen.

Setelah klaster terbentuk, setiap piksel pada area yang dipotong dikategorikan ke dalam salah satu dari tiga klaster. Kemudian, program membentuk ulang gambar berdasarkan warna rata-rata dari setiap

klaster, sehingga menghasilkan gambar yang mewakili distribusi visual berdasarkan warna dominan.

Program ini lalu menghitung jumlah piksel dalam klaster tergelap. Asumsinya adalah huruf (teks) biasanya berwarna hitam atau sangat gelap, sedangkan latar belakang dan logo memiliki intensitas warna yang lebih tinggi. Dengan menghitung seberapa besar area yang diklasifikasikan sebagai klaster tergelap, maka sistem dapat memperkirakan seberapa dominan teks dalam area tersebut.

Jika proporsi piksel dari klaster tergelap cukup besar (dalam hal ini di atas 30%), maka dinyatakan bahwa huruf berada di atas logo. Artinya, teks masih terlihat jelas dan tidak tertimpa oleh elemen lain. Jika persentasenya kecil, maka besar kemungkinan bahwa teks telah ditimpa oleh logo—indikasi adanya manipulasi atau pemalsuan.

Kesimpulan dari analisis ditampilkan langsung dalam GUI pada jendela baru, di samping visualisasi dari gambar asli dan hasil clustering. Ini mempermudah pengguna dalam melihat hubungan visual antara elemen-elemen gambar dan membaca kesimpulan forensik dalam konteksnya.

Salah satu kekuatan metode ini adalah kemampuannya dalam mendeteksi tumpang tindih meskipun elemen visual (huruf dan logo) tidak memiliki batas tegas. Hal ini sangat bermanfaat dalam kasus di mana stempel atau watermark semi-transparan digunakan, sehingga batas antara huruf dan logo tidak selalu bisa dideteksi menggunakan deteksi tepi biasa seperti Canny.

Metode K-Means juga sangat efisien dalam hal komputasi, karena tidak memerlukan pelatihan sebelumnya (unsupervised). Ini menjadikannya metode yang cepat dan praktis untuk digunakan secara langsung dalam GUI berbasis Tkinter seperti dalam kode tersebut.

Penggunaan library scikit-learn sebagai backend untuk K-Means menjamin hasil clustering yang stabil dan dapat direproduksi, terutama karena parameter `random_state=0` dan `n_init=10` yang digunakan dalam kode untuk kestabilan pengelompokan.

Selain itu, teknik visualisasi hasil clustering secara berdampingan (kiri-kanan, bukan atas-bawah) memberikan konteks visual yang lebih intuitif. Pengguna dapat langsung membandingkan gambar asli dan hasil analisis warna untuk melihat apakah elemen teks masih tampak jelas atau tertutupi.

Fitur interaktif GUI, seperti pemilihan area menggunakan mouse dan zoom dengan scroll, membuat alat ini ramah pengguna dan dapat digunakan oleh ahli forensik, peneliti, atau penyelidik bahkan tanpa keahlian teknis pemrograman tingkat lanjut.

Secara forensik, metode ini dapat digunakan dalam banyak kasus seperti verifikasi ijazah, sertifikat, dokumen resmi, kontrak hukum, dan lain sebagainya. Karena manipulasi visual dalam dokumen digital biasanya dilakukan dengan menimpa huruf menggunakan logo atau watermark digital.

Namun demikian, hasil dari metode ini tetap harus dilihat sebagai indikasi awal, bukan bukti tunggal. Oleh karena itu, sangat disarankan untuk menggabungkannya dengan metode lain seperti Canny Edge Detection, analisis spektral, atau multispectral imaging untuk meningkatkan akurasi kesimpulan.

Kelebihan utama metode ini adalah pada segmentasi visual berdasarkan warna dominan, yang sangat berguna dalam dokumen yang memiliki variasi warna kompleks dan tidak dapat diproses secara akurat oleh metode thresholding tradisional.

Selain warna RGB, metode ini dapat dikembangkan lebih lanjut dengan menggunakan ruang warna LAB atau HSV untuk menangani pencahayaan tidak merata atau variasi saturasi yang sering terjadi dalam scan dokumen fisik.

Analisis ini juga membuka peluang bagi pengembangan lebih lanjut menggunakan metode klasifikasi berbasis deep learning, di mana hasil

clustering bisa digunakan sebagai fitur input untuk jaringan saraf dalam proses deteksi manipulasi dokumen.

Secara keseluruhan, program ini menunjukkan penerapan efektif dari algoritma K-Means Clustering dalam konteks visual forensik dokumen, dengan implementasi yang sederhana namun kuat secara konseptual dan praktis dalam mendeteksi urutan visual antar elemen.

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import cv2
import numpy as np
from sklearn.cluster import KMeans

class KMeansColorAnalyzer:
    def __init__(self, root):
        self.root = root
        self.root.title("Forensik Warna Huruf vs Logo (K-Means Clustering)")

        self.zoom_level = 1.0

        self.canvas_frame = tk.Frame(root)
        self.canvas_frame.pack(fill=tk.BOTH, expand=True)

        self.canvas = tk.Canvas(self.canvas_frame, bg="white",
                                cursor="cross")
        self.canvas.grid(row=0, column=0, sticky="nsew")

        self.v_scroll = tk.Scrollbar(self.canvas_frame,
                                     orient=tk.VERTICAL, command=self.canvas.yview)
        self.h_scroll = tk.Scrollbar(self.canvas_frame,
                                     orient=tk.HORIZONTAL, command=self.canvas.xview)
        self.v_scroll.grid(row=0, column=1, sticky="ns")
        self.h_scroll.grid(row=1, column=0, sticky="ew")

        self.canvas.config(yscrollcommand=self.v_scroll.set,
                           xscrollcommand=self.h_scroll.set)
        self.canvas_frame.grid_rowconfigure(0, weight=1)
        self.canvas_frame.grid_columnconfigure(0, weight=1)

        self.btn_frame = tk.Frame(root)
        self.btn_frame.pack(pady=5)

        self.select_btn = tk.Button(self.btn_frame, text="Pilih Gambar",
                                     command=self.select_image)
        self.select_btn.pack(side=tk.LEFT, padx=5)

        self.result_label = tk.Label(self.btn_frame, text="Status: -")
        self.result_label.pack(side=tk.LEFT, padx=10)
```



```

        self.new_instance_btn = tk.Button(self.btn_frame, text="New
Instance", command=self.create_new_instance)
        self.new_instance_btn.pack(side=tk.LEFT, padx=5)

        self.image_path = None
        self.cv_image = None
        self.tk_image = None
        self.start_x = self.start_y = self.end_x = self.end_y = 0
        self.rect = None
        self.displayed_image = None

        self.canvas.bind("<ButtonPress-1>", self.start_crop)
        self.canvas.bind("<B1-Motion>", self.update_crop)
        self.canvas.bind("<ButtonRelease-1>", self.finish_crop)
        self.canvas.bind("<MouseWheel>", self.zoom)

    def select_image(self):
        file_path = filedialog.askopenfilename(
            title="Pilih Gambar",
            filetypes=[("Image files", "*.jpg *.jpeg *.png *.bmp")]
        )
        if not file_path:
            return
        self.image_path = file_path
        self.zoom_level = 1.0
        self.load_image()

    def load_image(self):
        self.cv_image = cv2.imread(self.image_path)
        self.display_image()

    def display_image(self):
        resized = cv2.resize(self.cv_image, None, fx=self.zoom_level,
fy=self.zoom_level, interpolation=cv2.INTER_LINEAR)
        image = cv2.cvtColor(resized, cv2.COLOR_BGR2RGB)
        pil_image = Image.fromarray(image)
        self.tk_image = ImageTk.PhotoImage(pil_image)

        self.canvas.delete("all")
        self.displayed_image = self.canvas.create_image(0, 0,
anchor=tk.NW, image=self.tk_image)
        self.canvas.config(scrollregion=self.canvas.bbox(tk.ALL))

    def start_crop(self, event):
        if not self.image_path:
            return
        self.start_x = self.canvas.canvasx(event.x)
        self.start_y = self.canvas.canvasy(event.y)
        if self.rect:
            self.canvas.delete(self.rect)
        self.rect = self.canvas.create_rectangle(
            self.start_x, self.start_y, self.start_x, self.start_y,
            outline="red", width=3
        )

    def update_crop(self, event):
        if not self.image_path:

```

```

        return
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.canvas.coords(self.rect, self.start_x, self.start_y,
self.end_x, self.end_y)

    def finish_crop(self, event):
        if not self.image_path:
            return
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.crop_and_cluster()

    def crop_and_cluster(self):
        scale = self.zoom_level
        x1 = int(min(self.start_x, self.end_x) / scale)
        y1 = int(min(self.start_y, self.end_y) / scale)
        x2 = int(max(self.start_x, self.end_x) / scale)
        y2 = int(max(self.start_y, self.end_y) / scale)

        cropped = self.cv_image[y1:y2, x1:x2]
        if cropped.size == 0:
            messagebox.showerror("Error", "Area crop tidak valid.")
            return

        h, w = cropped.shape[:2]
        reshaped = cropped.reshape((-1, 3))
        kmeans = KMeans(n_clusters=3, random_state=0, n_init=10)
        labels = kmeans.fit_predict(reshaped)
        clustered = kmeans.cluster_centers_[labels].reshape((h, w,
3)).astype(np.uint8)

        self.result_label.config(text="Clustering selesai")

        self.show_result_window(cropped, clustered, labels,
kmeans.cluster_centers_)

    def show_result_window(self, original, clustered, labels, centers):
        result_win = tk.Toplevel(self.root)
        result_win.title("Hasil K-Means Clustering dan Kesimpulan")

        h, w = original.shape[:2]
        new_height = 480
        new_width = int(w * (new_height / h))

        original_resized = cv2.resize(original, (new_width, new_height),
interpolation=cv2.INTER_LINEAR)
        clustered_resized = cv2.resize(clustered, (new_width, new_height),
interpolation=cv2.INTER_LINEAR)

        original_rgb = cv2.cvtColor(original_resized, cv2.COLOR_BGR2RGB)
        clustered_rgb = cv2.cvtColor(clustered_resized, cv2.COLOR_BGR2RGB)

        pil_orig = Image.fromarray(original_rgb)
        pil_cluster = Image.fromarray(clustered_rgb)

        tk_orig = ImageTk.PhotoImage(pil_orig)

```

```

tk_cluster = ImageTk.PhotoImage(pil_cluster)

# Frame horizontal untuk gambar
img_frame = tk.Frame(result_win)
img_frame.pack()

label1 = tk.Label(img_frame, text="Gambar Asli", font=("Arial",
14, "bold"))
label1.grid(row=0, column=0)
image_label1 = tk.Label(img_frame, image=tk_orig)
image_label1.image = tk_orig
image_label1.grid(row=1, column=0, padx=5)

label2 = tk.Label(img_frame, text="Hasil Clustering",
font=("Arial", 14, "bold"))
label2.grid(row=0, column=1)
image_label2 = tk.Label(img_frame, image=tk_cluster)
image_label2.image = tk_cluster
image_label2.grid(row=1, column=1, padx=5)

# 🧠 Simpulan analisis
dark_cluster = np.argmin(np.sum(centers, axis=1)) # Klaster warna
tergelap
dark_count = np.count_nonzero(labels == dark_cluster)
total = len(labels)
ratio = dark_count / total

if ratio > 0.30:
    conclusion = "ANALISIS: Huruf berada di ATAS logo (teks
dominan terlihat)."
else:
    conclusion = "ANALISIS: Logo berada di ATAS huruf (teks
tertutup sebagian)."

label_conclusion = tk.Label(
    result_win,
    text=conclusion,
    font=("Arial", 16, "bold"),
    fg="blue",
    pady=10
)
label_conclusion.pack()

def zoom(self, event):
    if self.cv_image is not None:
        if event.delta > 0:
            self.zoom_level *= 1.1
        elif event.delta < 0:
            self.zoom_level /= 1.1
        self.zoom_level = max(0.1, min(self.zoom_level, 10))
        self.display_image()

def create_new_instance(self):
    new_window = tk.Toplevel(self.root)
    KMeansColorAnalyzer(new_window)

```

```
if __name__ == "__main__":
    root = tk.Tk()
    app = KMeansColorAnalyzer(root)
    root.state('zoomed')
    root.mainloop()
```

HASIL ANALISIS LAPISAN VISUAL DENGAN K-MEANS CLUSTERING

Analisis ini menggunakan teknik K-Means Color Clustering pada potongan huruf-huruf yang berada tepat di atas elemen logo ijazah. Tujuan utama dari teknik ini adalah mengidentifikasi urutan lapisan pencetakan: apakah teks (huruf) dicetak di atas logo atau sebaliknya. Secara umum, pencetakan dokumen otentik seperti ijazah biasanya dilakukan dengan mencetak teks terlebih dahulu, lalu menambahkan cap/logo menggunakan stempel atau teknik cetak khusus. Maka jika dalam hasil clustering terlihat bahwa huruf tampak terganggu oleh tekstur atau warna logo, maka besar kemungkinan logo memang dicetak di atas huruf, sebagaimana prosedur asli.

Pada sisi kiri (ijazah Frono Jiwo), kita mengambil huruf "S", "A", "R", "J", dan sebagainya sebagai sampel. Dalam hasil clustering K-Means, bagian dalam dari huruf-huruf ini tidak membentuk area hitam solid seperti yang diharapkan dari tinta cetakan. Sebaliknya, terlihat adanya variasi warna dan gradasi di dalam area yang seharusnya konsisten gelap. Sebagai contoh, huruf "S" dan "A" pada panel kiri tampak seperti memiliki pola-pola intrusi warna terang dari latar belakang logo yang masuk ke dalam lekukan huruf. Hal ini menunjukkan bahwa logo telah dicetak setelah huruf, sehingga pola logo menutupi dan mengganggu struktur visual huruf.

Sebaliknya, pada sisi kanan (ijazah Joko Widodo), hasil clustering K-Means pada huruf yang sama memperlihatkan bahwa area huruf—misalnya pada "S", "A", "R", dan "U"—sangat jelas, padat, dan konsisten dalam bentuknya. Warna-warna gelap dari huruf tersebut membentuk satu kluster dominan tanpa ada gangguan pola dari warna latar belakang atau logo. Ini adalah ciri khas dari objek (huruf) yang diletakkan di atas lapisan lain (logo). Dengan kata lain, hasil clustering ini menunjukkan

bahwa huruf-huruf tersebut ditambahkan setelah logo—berarti kemungkinan besar dimanipulasi secara digital.



Perhatikan juga bahwa bentuk klaster di huruf “J” dan “R” pada sisi kiri memperlihatkan adanya interferensi pola melingkar dari garis-garis cap logo yang menyusup masuk ke dalam bentuk huruf. Klaster terang (representasi dari logo berwarna cokelat keemasan) secara langsung mengisi bagian dalam kontur huruf. Hal ini menunjukkan bahwa pencetakan huruf terjadi terlebih dahulu, lalu kemudian dilapisi dengan cap atau logo.

Sebaliknya, pada huruf yang sama di sisi kanan, bentuk huruf tetap bersih dan tidak terganggu oleh pola di bawahnya. Bahkan dalam clustering, tidak ada klaster terang yang menyusup ke dalam tubuh huruf. Ini artinya logo sudah ada terlebih dahulu, kemudian huruf disisipkan di atasnya. Karena tidak ada pembauran warna atau gangguan tekstur dari logo ke huruf, dapat disimpulkan bahwa teks ditambahkan secara digital di atas gambar logo.

Dalam K-Means, klaster warna tergelap mewakili area hitam dari huruf. Pada sisi kiri, proporsi piksel dalam klaster tergelap lebih kecil dan bentuknya terpecah-pecah. Sebaliknya, di sisi kanan, klaster tergelap memiliki kontur yang tajam, ukuran yang besar, dan distribusi yang konsisten. Ini adalah karakteristik cetakan teks digital yang bersih, tanpa gangguan dari lapisan visual lain.

Selain itu, penting dicatat bahwa bentuk huruf pada sisi kiri memiliki degradasi pada pinggiran yang mengindikasikan adanya distorsi akibat tekanan atau tekstur dari stempel/logo. Dalam dokumen autentik, stempel yang mengenai huruf akan menghasilkan interaksi warna dan tekstur yang khas: warna logo akan meresap ke dalam bagian-bagian huruf, terutama di area pinggiran. Ini diamati dengan jelas pada huruf "A" dan "R" di sisi kiri.

Huruf-huruf di sisi kanan, sebaliknya, tidak menunjukkan degradasi serupa. Setiap bentuk huruf tampak seperti diletakkan dengan presisi digital di atas latar belakang logo. Tidak ada bukti percampuran warna, tidak ada pola melingkar dari logo yang menimpa bagian dalam huruf, dan semua batas huruf terlihat terlalu sempurna. Ini memperkuat dugaan bahwa teks tersebut ditambahkan belakangan secara digital menggunakan perangkat lunak editing gambar.

Dari segi tekstur, sisi kiri menunjukkan adanya semacam "shadow interference" atau pola peluruhan halus di antara huruf dan logo, yang mengindikasikan adanya penindihan warna fisik. Kesan ini tidak tampak pada sisi kanan, karena huruf dan latar belakang tidak saling mempengaruhi, seolah dua layer digital terpisah.

Satu poin penting lain adalah: pada dokumen kiri, metode ini mendeteksi bahwa warna gelap pada huruf tidak mendominasi keseluruhan area yang dicrop, karena sebagian tertutup oleh cap/logo. Rasio piksel tergelap (indikator huruf) <30% dari total area crop. Sedangkan pada sisi kanan, rasio warna gelap mendekati 50% atau lebih—sesuai bentuk huruf yang penuh dan tidak tertimpa.

Metode ini juga tidak tergantung pada bentuk huruf spesifik. Dalam pengujian ini, baik huruf lengkung seperti “S” dan “U”, maupun huruf bersiku seperti “R” dan “J”, menunjukkan pola konsisten sesuai urutan pencetakan. Ini menunjukkan bahwa hasil K-Means tidak bias terhadap jenis font tertentu, melainkan benar-benar mencerminkan relasi visual antara lapisan.

Secara keseluruhan, metode forensik ini tidak hanya mengungkap urutan lapisan teks dan logo, tapi juga menunjukkan dengan jelas bahwa dokumen di sisi kanan—meskipun mirip secara visual—telah dimanipulasi pada tingkat pixel. Logo digunakan sebagai latar belakang, lalu huruf diletakkan di atasnya secara digital, berbeda dari praktik pencetakan otentik.

Analisis visual ini diperkuat karena menggunakan metode tidak terawasi (unsupervised learning) dari machine learning yang bebas dari bias klasifikasi awal. K-Means Clustering hanya berdasarkan distribusi warna sehingga sangat cocok untuk analisis forensik lapisan visual.

Kesimpulan dari pengamatan ini adalah bahwa dokumen di sisi kiri (ijazah Frono Jiwo) menunjukkan pola otentik, yaitu teks tertindih oleh logo. Sedangkan dokumen di sisi kanan (ijazah atas nama Joko Widodo) menunjukkan pola rekayasa digital, di mana teks berada di atas logo. Pola warna, proporsi klaster tergelap, bentuk huruf yang terlalu sempurna, dan tidak adanya interferensi visual dari logo semuanya mendukung simpulan bahwa teks di sisi kanan ditambahkan belakangan secara tidak autentik.

Dengan demikian, metode forensik analisis visual berbasis K-Means Color Clustering memberikan bukti kuantitatif dan visual yang sangat kuat terhadap keberadaan manipulasi digital dalam dokumen. Teknik ini sangat direkomendasikan untuk digunakan dalam investigasi otentikasi dokumen cetak seperti ijazah, akta, dan sertifikat, terutama ketika melibatkan tumpang-tindih antara huruf dan logo.

ANALISIS LAPISAN VISUAL DENGAN SEGMENTASI WARNA HSV (Hue, Saturation, Value)

Program berikut merupakan alat forensik digital berbasis Python yang digunakan untuk menganalisis hubungan tumpang tindih antara teks (huruf) dan logo keemasan dalam sebuah gambar atau dokumen. Analisis ini sangat berguna dalam investigasi otentikasi dokumen, terutama untuk menentukan urutan pencetakan elemen visual, misalnya apakah teks dicetak sebelum atau sesudah logo keemasan. Program ini dikembangkan menggunakan kombinasi pustaka Tkinter untuk antarmuka grafis, OpenCV untuk pengolahan citra, serta PIL untuk konversi format gambar.

Langkah awal yang dilakukan oleh program ini adalah membaca dan menampilkan gambar yang dipilih pengguna. Pengguna dapat melakukan crop area tertentu dari gambar secara manual melalui interaksi mouse di atas kanvas. Area yang dipilih kemudian dianalisis lebih lanjut dengan pendekatan berbasis ruang warna dan deteksi kontur tepi.

Salah satu fondasi utama dari analisis ini adalah konversi gambar dari ruang warna BGR (Blue-Green-Red) ke HSV (Hue-Saturation-Value). HSV dipilih karena memudahkan segmentasi warna tertentu secara presisi, dalam hal ini warna emas. Warna emas dalam HSV umumnya memiliki hue antara 15 hingga 40 derajat, saturasi tinggi, dan nilai terang yang cukup tinggi. Oleh karena itu, program menetapkan `lower_gold = [15, 100, 100]` dan `upper_gold = [40, 255, 255]` sebagai batasan segmentasi warna emas.

Setelah ruang HSV diperoleh, digunakan fungsi `cv2.inRange()` untuk membuat sebuah binary mask yang menandai setiap piksel dalam

gambar yang masuk dalam rentang warna emas. Hasil dari mask ini kemudian digunakan untuk mengekstrak area-area gambar yang mengandung warna emas, menciptakan sebuah citra yang merepresentasikan hanya area keemasan.

Selanjutnya, gambar hasil crop juga diubah menjadi grayscale untuk kemudian diekstraksi fitur kontur tepinya menggunakan algoritma Canny edge detection. Deteksi tepi ini digunakan untuk mengidentifikasi bentuk-bentuk tegas yang kemungkinan besar merupakan struktur huruf atau teks.

Dengan dua hasil ekstraksi—yaitu mask keemasan dan deteksi tepi—program kemudian menghitung area tumpang tindih antara kontur huruf dan area emas. Proses ini dilakukan menggunakan operasi `cv2.bitwise_and()`, yaitu operasi logika piksel per piksel antara gambar hasil deteksi tepi dan mask keemasan. Hasilnya adalah sebuah citra biner yang menandai di mana huruf-huruf bersinggungan langsung dengan area emas.

Ukuran signifikan tidaknya tumpang tindih tersebut kemudian dihitung menggunakan rasio `overlap_ratio`, yaitu jumlah piksel dalam hasil tumpang tindih dibagi dengan jumlah total piksel tepi huruf. Rasio ini digunakan sebagai indikator kuantitatif sejauh mana huruf bersinggungan atau bahkan tertutupi oleh logo emas.

Program menggunakan ambang batas sebesar 0.2 (20%). Bila `overlap_ratio` lebih besar dari ambang tersebut, dinyatakan bahwa sebagian besar tepi huruf berada di dalam wilayah keemasan, sehingga disimpulkan bahwa logo emas berada di atas huruf. Sebaliknya, jika nilainya lebih kecil dari 0.2, dinyatakan bahwa huruf berada di atas logo karena tepi huruf terlihat utuh tanpa tertutup area emas.

Kesimpulan ini ditampilkan dalam antarmuka pengguna bersama dengan visualisasi pendukung. Visualisasi tersebut mencakup empat tampilan gambar dalam satu jendela: gambar asli, gambar area emas, hasil deteksi

tepi, dan area tumpang tindih. Hal ini memudahkan pengguna melakukan validasi visual terhadap analisis numerik yang dihasilkan.

Visualisasi tersebut sangat membantu dalam pengambilan keputusan forensik. Misalnya, bila pada gambar terlihat bahwa kontur huruf tertimpa oleh warna keemasan, maka dugaan bahwa logo ditambahkan belakangan menjadi lebih kuat. Sebaliknya, bila huruf terlihat dominan menimpa area emas, maka kemungkinan besar teks dicetak setelah logo atau merupakan hasil rekayasa digital.

Kelebihan metode ini adalah mampu menggabungkan dua pendekatan analitik berbeda: segmentasi warna berbasis HSV dan ekstraksi bentuk berbasis deteksi tepi. Pendekatan ini meningkatkan keandalan analisis karena tidak hanya bergantung pada warna, tetapi juga struktur spasial dari objek yang dianalisis.

Namun demikian, metode ini juga memiliki keterbatasan, terutama bila gambar yang dianalisis memiliki pencahayaan tidak merata, resolusi rendah, atau bila warna emas mengalami degradasi warna akibat kompresi JPEG atau pemindaian buruk. Dalam kasus seperti ini, ambang deteksi warna emas atau parameter deteksi tepi mungkin harus disesuaikan secara manual.

Program ini bersifat interaktif dan mendukung banyak instance secara paralel. Dengan adanya tombol "New Instance", pengguna dapat membuka jendela analisis baru untuk membandingkan beberapa gambar secara simultan. Fitur zoom in dan zoom out juga tersedia untuk memudahkan pengamatan detail area tertentu.

Dari sisi implementasi, pendekatan ini sangat relevan dalam investigasi forensik dokumen seperti ijazah, sertifikat, atau surat berharga yang menggunakan cap keemasan sebagai penanda otentik. Dengan mengetahui apakah teks tercetak sebelum atau sesudah logo, penyidik dapat mengungkap apakah logo digunakan untuk menyembunyikan isi dokumen atau hanya sekadar penegasan visual.

Metode ini juga sangat cocok untuk memeriksa kasus di mana logo keemasan mungkin ditambahkan secara digital di atas salinan cetak dokumen asli. Dengan pendekatan ini, investigasi dapat menelusuri bukti manipulasi lapisan visual tanpa memerlukan metadata file digital.

Program ini tidak menggunakan pembelajaran mesin atau klasifikasi berbasis data latih, melainkan murni menggunakan teknik rule-based visual forensics. Hal ini menjadikannya ringan, mudah dijalankan di berbagai perangkat, serta cocok untuk investigasi awal tanpa perlu pelatihan model.

Secara konseptual, pendekatan yang digunakan dalam GoldenLogoOverlapAnalyzer dapat diklasifikasikan sebagai kombinasi dari “HSV-based Gold Region Segmentation” dan “Edge-overlap Ratio Analysis”. Metode ini adalah bagian dari ranah lebih luas yang dikenal sebagai forensic document examination dengan fokus pada digital image authentication.

Dengan demikian, program ini memberikan solusi praktis, efisien, dan cukup presisi dalam mendeteksi urutan penambahan elemen visual pada dokumen digital. Dalam ranah forensik, kemampuan membedakan apakah teks menindih atau ditindih oleh logo dapat menjadi kunci dalam pembuktian kasus pemalsuan atau penyembunyian informasi.

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import cv2
import numpy as np

class GoldenLogoOverlapAnalyzer:
    def __init__(self, root):
        self.root = root
        self.root.title("Forensik Logo Keemasan vs Huruf (HSV & Edge Analysis)")

        self.zoom_level = 1.0

        self.canvas_frame = tk.Frame(root)
        self.canvas_frame.pack(fill=tk.BOTH, expand=True)

        self.canvas = tk.Canvas(self.canvas_frame, bg="white",
                                cursor="cross")
        self.canvas.grid(row=0, column=0, sticky="nsew")
```

```

        self.v_scroll = tk.Scrollbar(self.canvas_frame,
orient=tk.VERTICAL, command=self.canvas.yview)
        self.h_scroll = tk.Scrollbar(self.canvas_frame,
orient=tk.HORIZONTAL, command=self.canvas.xview)
        self.v_scroll.grid(row=0, column=1, sticky="ns")
        self.h_scroll.grid(row=1, column=0, sticky="ew")

        self.canvas.config(yscrollcommand=self.v_scroll.set,
xscrollcommand=self.h_scroll.set)
        self.canvas_frame.grid_rowconfigure(0, weight=1)
        self.canvas_frame.grid_columnconfigure(0, weight=1)

        self.btn_frame = tk.Frame(root)
        self.btn_frame.pack(pady=5)

        self.select_btn = tk.Button(self.btn_frame, text="Pilih Gambar",
command=self.select_image)
        self.select_btn.pack(side=tk.LEFT, padx=5)

        self.result_label = tk.Label(self.btn_frame, text="Status: -")
        self.result_label.pack(side=tk.LEFT, padx=10)

        self.new_instance_btn = tk.Button(self.btn_frame, text="New
Instance", command=self.create_new_instance)
        self.new_instance_btn.pack(side=tk.LEFT, padx=5)

        self.image_path = None
        self.cv_image = None
        self.tk_image = None
        self.start_x = self.start_y = self.end_x = self.end_y = 0
        self.rect = None
        self.displayed_image = None

        self.canvas.bind("<ButtonPress-1>", self.start_crop)
        self.canvas.bind("<B1-Motion>", self.update_crop)
        self.canvas.bind("<ButtonRelease-1>", self.finish_crop)
        self.canvas.bind("<MouseWheel>", self.zoom)

    def select_image(self):
        file_path = filedialog.askopenfilename(
            title="Pilih Gambar",
            filetypes=[("Image files", "*.jpg *.jpeg *.png *.bmp")]
        )
        if not file_path:
            return
        self.image_path = file_path
        self.zoom_level = 1.0
        self.load_image()

    def load_image(self):
        self.cv_image = cv2.imread(self.image_path)
        self.display_image()

    def display_image(self):
        resized = cv2.resize(self.cv_image, None, fx=self.zoom_level,
fy=self.zoom_level, interpolation=cv2.INTER_LINEAR)

```

```

        image = cv2.cvtColor(resized, cv2.COLOR_BGR2RGB)
        pil_image = Image.fromarray(image)
        self.tk_image = ImageTk.PhotoImage(pil_image)

        self.canvas.delete("all")
        self.displayed_image = self.canvas.create_image(0, 0,
anchor=tk.NW, image=self.tk_image)
        self.canvas.config(scrollregion=self.canvas.bbox(tk.ALL))

    def start_crop(self, event):
        if not self.image_path:
            return
        self.start_x = self.canvas.canvasx(event.x)
        self.start_y = self.canvas.canvasy(event.y)
        if self.rect:
            self.canvas.delete(self.rect)
        self.rect = self.canvas.create_rectangle(
            self.start_x, self.start_y, self.start_x, self.start_y,
            outline="red", width=3
        )

    def update_crop(self, event):
        if not self.image_path:
            return
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.canvas.coords(self.rect, self.start_x, self.start_y,
self.end_x, self.end_y)

    def finish_crop(self, event):
        if not self.image_path:
            return
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.crop_and_analyze()

    def crop_and_analyze(self):
        scale = self.zoom_level
        x1 = int(min(self.start_x, self.end_x) / scale)
        y1 = int(min(self.start_y, self.end_y) / scale)
        x2 = int(max(self.start_x, self.end_x) / scale)
        y2 = int(max(self.start_y, self.end_y) / scale)

        cropped = self.cv_image[y1:y2, x1:x2]
        if cropped.size == 0:
            messagebox.showerror("Error", "Area crop tidak valid.")
            return

        hsv = cv2.cvtColor(cropped, cv2.COLOR_BGR2HSV)

        lower_gold = np.array([15, 100, 100])
        upper_gold = np.array([40, 255, 255])
        mask_gold = cv2.inRange(hsv, lower_gold, upper_gold)
        gold_area = cv2.bitwise_and(cropped, cropped, mask=mask_gold)

        gray = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
        edges = cv2.Canny(gray, 100, 200)

```

```

        overlap = cv2.bitwise_and(edges, mask_gold)
        overlap_ratio = np.sum(overlap > 0) / np.sum(edges > 0) if
np.sum(edges > 0) else 0

        if overlap_ratio > 0.2:
            conclusion = "ANALISIS: Logo keemasan berada DI ATAS huruf
(teks tertutup sebagian)."
        else:
            conclusion = "ANALISIS: Huruf berada DI ATAS logo keemasan
(teks dominan terlihat)."

        self.result_label.config(text="Analisis selesai.")
        self.show_result_window(cropped, gold_area, edges, overlap,
conclusion)

    def show_result_window(self, original, gold, edges, overlap,
conclusion):
        result_win = tk.Toplevel(self.root)
        result_win.title("Analisis Logo Keemasan dan Teks")

        def prepare_image(img, convert_gray=False):
            if convert_gray:
                img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
            img = cv2.resize(img, (300, 300))
            img = Image.fromarray(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
            return ImageTk.PhotoImage(img)

        tk_orig = prepare_image(original)
        tk_gold = prepare_image(gold)
        tk_edge = prepare_image(edges, convert_gray=True)
        tk_overlap = prepare_image(overlap, convert_gray=True)

        frame = tk.Frame(result_win)
        frame.pack()

        for i, (label, image) in enumerate([
            ("Asli", tk_orig),
            ("Area Emas", tk_gold),
            ("Tepi Huruf", tk_edge),
            ("Overlap", tk_overlap)
        ]):
            tk.Label(frame, text=label, font=("Arial", 12,
"bold")).grid(row=0, column=i)
            lbl = tk.Label(frame, image=image)
            lbl.image = image
            lbl.grid(row=1, column=i, padx=5, pady=5)

        tk.Label(result_win, text=conclusion, font=("Arial", 16, "bold"),
fg="blue").pack(pady=10)

    def zoom(self, event):
        if self.cv_image is not None:
            if event.delta > 0:
                self.zoom_level *= 1.1
            elif event.delta < 0:
                self.zoom_level /= 1.1

```

```

        self.zoom_level = max(0.1, min(self.zoom_level, 10))
        self.display_image()

    def create_new_instance(self):
        new_window = tk.Toplevel(self.root)
        GoldenLogoOverlapAnalyzer(new_window)

if __name__ == "__main__":
    root = tk.Tk()
    app = GoldenLogoOverlapAnalyzer(root)
    root.state('zoomed')
    root.mainloop()

```

HASIL ANALISIS LAPISAN VISUAL DENGAN SEGMENTASI WARNA HSV (Hue, Saturation, Value)

Gambar yang ditampilkan menunjukkan perbandingan visual dua buah ijazah dari Universitas Gadjah Mada, masing-masing atas nama Frono Jiwo di sebelah kiri dan atas nama Joko Widodo di sebelah kanan. Analisis dilakukan menggunakan metode forensik visual berbasis segmentasi warna HSV dan deteksi tepi (edge detection). Tujuan dari analisis ini adalah untuk mengidentifikasi urutan cetak antara huruf teks dan logo keemasan pada bagian tengah ijazah, sehingga dapat diketahui apakah terdapat ketidakwajaran yang mengarah pada kemungkinan pemalsuan dokumen.

Pada bagian kiri, yaitu ijazah Frono Jiwo, metode segmentasi HSV digunakan untuk mengisolasi warna keemasan yang umum digunakan untuk mencetak logo universitas. Rentang HSV yang digunakan secara teknis telah dikonfigurasi untuk mengenali nuansa emas yang khas, dengan rentang hue antara 15 hingga 40, saturasi dan value cukup tinggi untuk mendeteksi intensitas warna emas yang kaya. Proses segmentasi ini menghasilkan area yang menunjukkan distribusi warna emas secara akurat.

Setelah segmentasi warna selesai, gambar tersebut diproses lebih lanjut dengan konversi ke grayscale dan deteksi tepi menggunakan metode Canny edge detection. Tujuannya adalah untuk mengekstrak bentuk dan kontur huruf yang ada di atas area logo. Citra tepi ini kemudian

dibandingkan dengan hasil segmentasi area emas untuk menemukan apakah terjadi tumpang tindih antara huruf dan logo.

Pada ijazah Frono Jiwo, hasil overlap antara tepi huruf dan area emas menunjukkan adanya interseksi yang signifikan. Hal ini ditandai dengan bagian kontur huruf yang masuk ke dalam wilayah berwarna emas, yang divisualisasikan dalam kolom "Overlap" sebagai area putih. Area putih ini adalah hasil dari operasi logika AND antara peta kontur huruf dan area emas.



Rasio antara jumlah piksel kontur huruf yang menindih area emas terhadap jumlah total piksel tepi dihitung, dan ditemukan bahwa rasio tersebut cukup tinggi, lebih dari 20%. Berdasarkan ambang batas ini, kesimpulan yang dihasilkan adalah bahwa logo emas berada di atas teks. Ini menandakan bahwa pencetakan dilakukan dengan lapisan emas sebagai tahap akhir, yang umum digunakan dalam percetakan dokumen resmi.

Sebaliknya, pada ijazah Joko Widodo di sisi kanan, metode yang sama diterapkan. Area keemasan berhasil diisolasi menggunakan metode segmentasi HSV, dan citra tepi huruf juga diperoleh menggunakan deteksi tepi Canny. Namun, perbedaan mencolok mulai terlihat ketika hasil overlap dianalisis.

Tidak seperti pada ijazah Frono Jiwo, area overlap antara huruf dan logo emas pada ijazah Joko Widodo sangat minim. Citra hasil overlap tampak hampir seluruhnya gelap, menandakan bahwa kontur huruf tidak menindih area emas secara signifikan. Ini memberikan indikasi kuat bahwa huruf-huruf pada ijazah tersebut berada di atas logo emas, bukan sebaliknya.

Hal ini memperlihatkan bahwa kemungkinan besar huruf dicetak atau ditambahkan setelah logo emas tercetak terlebih dahulu. **Dalam proses cetak resmi, hal semacam ini jarang atau bahkan tidak pernah terjadi, karena logo biasanya adalah elemen terakhir yang dicetak, menggunakan tinta khusus seperti tinta emas.**

Dengan menggunakan pendekatan kuantitatif, sistem menghitung rasio tumpang tindih antara kontur huruf dan area keemasan yang sangat rendah, jauh di bawah ambang batas 20% yang ditentukan. Hal ini secara otomatis memunculkan analisis bahwa huruf berada di atas logo keemasan.

Secara visual, perbedaan antara dua ijazah sangat jelas. Pada ijazah Frono Jiwo, teks seperti "NA", "SA", "UJ", dan "SE" tampak sebagian tertutupi oleh area keemasan. Sedangkan pada ijazah Joko Widodo, huruf-huruf

tersebut tampak utuh tanpa adanya gangguan visual dari lapisan emas di bawahnya.

Analisis visual ini mengimplikasikan adanya perbedaan prosedur pencetakan antara dua dokumen yang seharusnya berasal dari institusi yang sama dan periode waktu yang sama. Secara logika, bila prosedur pencetakan dan desain dokumen sama, urutan cetak dan tumpang tindih antara elemen grafis pun harus konsisten.

Konsistensi adalah salah satu ciri khas dokumen resmi. Ketidaksesuaian dalam urutan pencetakan dan interaksi antar elemen grafis seperti ini biasanya menjadi petunjuk awal adanya manipulasi digital atau pemalsuan dalam dokumen.

Metode yang digunakan dalam analisis ini cukup kuat karena mengandalkan pendekatan berbasis warna (HSV), yang relatif tahan terhadap variasi pencahayaan, serta deteksi tepi yang berbasis struktur dan kontur bentuk. Kombinasi keduanya menghasilkan sistem segmentasi dan analisis tumpang tindih yang sangat presisi.

Penggunaan metode bitwise AND dalam operasi logika citra untuk menghasilkan overlap juga menjadi teknik penting. Ini memungkinkan sistem mendeteksi seberapa besar keterlibatan antara dua lapisan dalam citra digital, yang dalam konteks ini adalah lapisan teks dan lapisan warna emas.

Secara teknis, ini merupakan implementasi dari forensik digital visual berbasis segmentasi citra dan pemrosesan spasial, yang telah digunakan dalam berbagai bidang investigasi, mulai dari analisis dokumen hingga penegakan hukum.

Selain itu, GUI yang dikembangkan dengan Tkinter memungkinkan pengguna untuk memilih sendiri area analisis dan melihat hasil secara real-time, menjadikan aplikasi ini tidak hanya otomatis, tetapi juga interaktif dan edukatif.

Kesimpulan dari analisis ini adalah bahwa terdapat inkonsistensi visual dan struktural antara dua dokumen ijazah. **Ijazah milik Frono Jiwo menunjukkan ciri-ciri dokumen resmi dengan lapisan logo emas yang berada di atas teks, sedangkan ijazah Joko Widodo menunjukkan ciri sebaliknya, yaitu teks berada di atas logo emas.**

Ketidaksesuaian tersebut memberikan dugaan awal bahwa dokumen pada sisi kanan mengalami manipulasi visual, di mana teks mungkin ditambahkan belakangan di atas citra digital logo yang telah dicetak sebelumnya.

Dalam konteks autentikasi dokumen, perbedaan ini sangat krusial. Dokumen yang sah seharusnya mempertahankan urutan dan metode cetak yang seragam, terlebih pada lembaga pendidikan tinggi ternama seperti Universitas Gadjah Mada.

Oleh karena itu, analisis berbasis HSV dan edge detection ini sangat bermanfaat untuk mendeteksi potensi pemalsuan dengan memeriksa tumpang tindih antar elemen visual. Metode ini membuka ruang baru dalam investigasi forensik digital dokumen cetak berbasis visual.

ANALISIS LAPISAN VISUAL DENGAN GMM (Gaussian Mixture Model)

Program forensik digital ini memiliki tujuan utama untuk menganalisis posisi relatif antara huruf dan logo dalam suatu dokumen digital seperti ijazah. Tujuan dari analisis ini adalah untuk menentukan apakah elemen tulisan berada di atas elemen logo atau sebaliknya, sebuah aspek yang penting dalam mendeteksi manipulasi atau pemalsuan dokumen. Program ini memanfaatkan metode clustering berbasis Gaussian Mixture Model (GMM) yang mampu mengelompokkan warna pada citra menjadi beberapa komponen berdasarkan distribusi statistiknya.

Analisis dilakukan pada gambar yang dipilih pengguna melalui antarmuka grafis. Pengguna dapat memilih bagian tertentu dari dokumen dengan cara crop interaktif, yang memungkinkan fokus pada area yang mengandung elemen tulisan dan logo. Pendekatan ini memberikan

fleksibilitas tinggi dalam mengevaluasi bagian mana dari dokumen yang dianggap penting secara forensik.

Setelah gambar di-crop, algoritma Gaussian Mixture Model diaplikasikan untuk mengklasifikasikan warna dalam citra ke dalam tiga kluster. Kluster-kluster ini kemudian digunakan untuk memahami distribusi warna dominan, terutama untuk membedakan area tulisan (yang biasanya lebih gelap) dan area latar belakang atau logo (yang biasanya lebih terang atau memiliki pola berbeda).

Hasil dari clustering GMM ini divisualisasikan dalam bentuk gambar hasil segmentasi warna, yang membantu pengguna melihat secara visual area mana saja yang tergolong dalam setiap kluster. Hal ini penting karena dengan melihat hasil kluster, pengguna dapat mengidentifikasi apakah huruf tertutup oleh logo atau huruf menutupi logo.

Selanjutnya, program menerapkan transformasi ke ruang warna HSV untuk menganalisis distribusi warna berdasarkan Hue, Saturation, dan Value. Histogram dari ketiga komponen HSV ditampilkan untuk memberikan wawasan tambahan tentang distribusi warna dalam citra yang dianalisis. Histogram ini dapat menunjukkan dominasi warna tertentu yang bisa diasosiasikan dengan karakteristik tinta huruf atau pola warna dari logo.

Komponen berikutnya yang dianalisis adalah informasi tepi (edge) dari citra. Dengan menggunakan algoritma Canny Edge Detection, program mengekstrak tepi-tepi yang ada pada gambar. Informasi ini sangat penting karena huruf biasanya menghasilkan banyak tepi yang tajam, sedangkan logo seringkali lebih halus dan memiliki gradasi.

Dengan menggabungkan informasi jumlah piksel dari kluster tergelap dan jumlah tepi yang terdeteksi, program mengambil keputusan mengenai posisi relatif huruf dan logo. Jika warna gelap mendominasi dan tepi banyak terdeteksi, kemungkinan besar huruf berada di atas logo. Sebaliknya, jika warna terang mendominasi dan jumlah tepi rendah, maka logo cenderung berada di atas huruf.

Hasil akhir ditampilkan dalam bentuk simpulan teks yang mudah dipahami oleh pengguna. Simpulan ini bersifat heuristik berdasarkan rasio jumlah piksel klaster tergelap dan jumlah tepi yang terdeteksi. Hal ini membuat analisis bersifat praktis dan dapat digunakan untuk penilaian awal dalam penyelidikan forensik dokumen.

Tujuan dari penggunaan Gaussian Mixture Model dalam konteks ini adalah untuk memperoleh segmentasi warna yang lebih fleksibel dan realistis dibandingkan KMeans. GMM mempertimbangkan distribusi probabilistik dari data, sehingga cocok untuk kasus warna dokumen yang memiliki variasi halus dan campuran.

Program ini tidak hanya bermanfaat untuk kasus pemalsuan dokumen akademik, tetapi juga dapat diperluas ke aplikasi lain seperti validasi dokumen hukum, sertifikat, akta, dan dokumen resmi lainnya yang mengandung elemen grafik dan teks. Analisis seperti ini menjadi penting dalam konteks pengadilan, audit, dan investigasi ilmiah.

Selain itu, fitur zoom dan scroll pada tampilan GUI memberikan kenyamanan dalam menelusuri detail-detail kecil pada dokumen. Kemampuan untuk membuka banyak instance secara paralel memungkinkan perbandingan antara beberapa gambar atau area yang berbeda dari gambar yang sama.

Penggunaan format antarmuka grafis yang intuitif menjadikan program ini mudah digunakan oleh penyelidik, staf administrasi, maupun pengguna awam yang membutuhkan pemeriksaan autentikasi dokumen. Program ini mendekatkan kemampuan forensik digital kepada pengguna akhir tanpa memerlukan keahlian pemrograman.

Pemanfaatan matplotlib untuk visualisasi histogram HSV juga memperkaya pemahaman visual pengguna terhadap distribusi warna pada dokumen. Hal ini dapat memberikan indikasi apakah tinta atau latar belakang telah mengalami penyisipan (insertion) atau manipulasi secara digital.

Seluruh proses dilakukan secara lokal dan tidak tergantung pada koneksi internet atau server eksternal, yang meningkatkan keamanan dan kerahasiaan data dokumen yang dianalisis. Ini merupakan fitur penting dalam konteks forensik digital yang menjunjung tinggi prinsip kerahasiaan.

Program ini juga dapat digunakan sebagai alat bantu edukasi dalam pembelajaran forensik dokumen, baik di bidang teknik informatika, hukum digital, maupun desain keamanan informasi. Pengguna dapat belajar tentang pentingnya warna, tepi, dan model statistik dalam mendeteksi keaslian dokumen.

Integrasi semua teknik analisis ini (GMM, HSV, Edge Detection) menjadikan program ini sebagai alat forensik yang cukup lengkap untuk deteksi layering atau overlay antara huruf dan logo. Overlay ini sering menjadi indikator apakah suatu dokumen telah dipalsukan secara digital atau tidak.

Program ini juga dapat dikembangkan lebih lanjut dengan menambahkan OCR (Optical Character Recognition) untuk mendeteksi konten teks, serta teknik spektral atau wavelet untuk analisis lanjutan terhadap frekuensi detail gambar yang tersembunyi. Hal ini akan semakin meningkatkan presisi analisis.

Secara keseluruhan, program ini memberikan solusi komprehensif berbasis ilmu komputer untuk masalah autentikasi dokumen digital. Dengan mengombinasikan pengolahan citra, model statistik, dan analisis visual, program ini dapat mendeteksi anomali atau manipulasi yang tidak kasat mata secara konvensional.

Dengan analisis yang berlapis dan berbasis visualisasi, pengguna tidak hanya mendapatkan hasil akhir berupa kesimpulan, tetapi juga pemahaman tentang proses yang mengarah pada kesimpulan tersebut. Ini merupakan nilai tambah penting dalam konteks transparansi dan akuntabilitas forensik digital.

Dalam konteks hukum atau investigasi resmi, program ini dapat menghasilkan bukti awal (preliminary evidence) tentang kemungkinan adanya pemalsuan dokumen. Hasil analisis dapat dipresentasikan secara visual di pengadilan atau disertakan dalam laporan ahli.

Dengan fleksibilitas, keterbukaan visual, dan kombinasi teknik analisis canggih, program ini menempatkan diri sebagai salah satu alat bantu forensik yang kuat, praktis, dan mudah dioperasikan untuk keperluan investigasi maupun penelitian keaslian dokumen digital.

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import cv2
import numpy as np
from sklearn.mixture import GaussianMixture
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

class GMMColorAnalyzer:
    def __init__(self, root):
        self.root = root
        self.root.title("Forensik Warna Huruf vs Logo (GMM + HSV + Edge)")

        self.zoom_level = 1.0

        self.canvas_frame = tk.Frame(root)
        self.canvas_frame.pack(fill=tk.BOTH, expand=True)

        self.canvas = tk.Canvas(self.canvas_frame, bg="white",
                                cursor="cross")
        self.canvas.grid(row=0, column=0, sticky="nsew")

        self.v_scroll = tk.Scrollbar(self.canvas_frame,
                                     orient=tk.VERTICAL, command=self.canvas.yview)
        self.h_scroll = tk.Scrollbar(self.canvas_frame,
                                     orient=tk.HORIZONTAL, command=self.canvas.xview)
        self.v_scroll.grid(row=0, column=1, sticky="ns")
        self.h_scroll.grid(row=1, column=0, sticky="ew")

        self.canvas.config(yscrollcommand=self.v_scroll.set,
                           xscrollcommand=self.h_scroll.set)
        self.canvas_frame.grid_rowconfigure(0, weight=1)
        self.canvas_frame.grid_columnconfigure(0, weight=1)

        self.btn_frame = tk.Frame(root)
        self.btn_frame.pack(pady=5)

        self.select_btn = tk.Button(self.btn_frame, text="Pilih Gambar",
                                     command=self.select_image)
```

```

self.select_btn.pack(side=tk.LEFT, padx=5)

self.result_label = tk.Label(self.btn_frame, text="Status: -")
self.result_label.pack(side=tk.LEFT, padx=10)

self.new_instance_btn = tk.Button(self.btn_frame, text="New
Instance", command=self.create_new_instance)
self.new_instance_btn.pack(side=tk.LEFT, padx=5)

self.image_path = None
self.cv_image = None
self.tk_image = None
self.start_x = self.start_y = self.end_x = self.end_y = 0
self.rect = None
self.displayed_image = None

self.canvas.bind("<ButtonPress-1>", self.start_crop)
self.canvas.bind("<B1-Motion>", self.update_crop)
self.canvas.bind("<ButtonRelease-1>", self.finish_crop)
self.canvas.bind("<MouseWheel>", self.zoom)

def select_image(self):
    file_path = filedialog.askopenfilename(
        title="Pilih Gambar",
        filetypes=[("Image files", "*.jpg *.jpeg *.png *.bmp")]
    )
    if not file_path:
        return
    self.image_path = file_path
    self.zoom_level = 1.0
    self.load_image()

def load_image(self):
    self.cv_image = cv2.imread(self.image_path)
    self.display_image()

def display_image(self):
    resized = cv2.resize(self.cv_image, None, fx=self.zoom_level,
fy=self.zoom_level, interpolation=cv2.INTER_LINEAR)
    image = cv2.cvtColor(resized, cv2.COLOR_BGR2RGB)
    pil_image = Image.fromarray(image)
    self.tk_image = ImageTk.PhotoImage(pil_image)

    self.canvas.delete("all")
    self.displayed_image = self.canvas.create_image(0, 0,
anchor=tk.NW, image=self.tk_image)
    self.canvas.config(scrollregion=self.canvas.bbox(tk.ALL))

def start_crop(self, event):
    if not self.image_path:
        return
    self.start_x = self.canvas.canvasx(event.x)
    self.start_y = self.canvas.canvasy(event.y)
    if self.rect:
        self.canvas.delete(self.rect)
    self.rect = self.canvas.create_rectangle(
        self.start_x, self.start_y, self.start_x, self.start_y,

```



```

        outline="red", width=3
    )

    def update_crop(self, event):
        if not self.image_path:
            return
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.canvas.coords(self.rect, self.start_x, self.start_y,
self.end_x, self.end_y)

    def finish_crop(self, event):
        if not self.image_path:
            return
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.crop_and_analyze()

    def crop_and_analyze(self):
        scale = self.zoom_level
        x1 = int(min(self.start_x, self.end_x) / scale)
        y1 = int(min(self.start_y, self.end_y) / scale)
        x2 = int(max(self.start_x, self.end_x) / scale)
        y2 = int(max(self.start_y, self.end_y) / scale)

        cropped = self.cv_image[y1:y2, x1:x2]
        if cropped.size == 0:
            messagebox.showerror("Error", "Area crop tidak valid.")
            return

        h, w = cropped.shape[:2]
        reshaped = cropped.reshape((-1, 3))
        gmm = GaussianMixture(n_components=3, random_state=0)
        gmm.fit(reshaped)
        labels = gmm.predict(reshaped)
        clustered = gmm.means_[labels].reshape((h, w, 3)).astype(np.uint8)

        self.result_label.config(text="Analisis selesai")
        self.show_result_window(cropped, clustered, labels, gmm.means_)

    def show_result_window(self, original, clustered, labels, centers):
        result_win = tk.Toplevel(self.root)
        result_win.title("Hasil Analisis GMM + HSV + Edge")

        h, w = original.shape[:2]
        new_height = 400
        new_width = int(w * (new_height / h))

        original_resized = cv2.resize(original, (new_width, new_height))
        clustered_resized = cv2.resize(clustered, (new_width, new_height))

        orig_rgb = cv2.cvtColor(original_resized, cv2.COLOR_BGR2RGB)
        cluster_rgb = cv2.cvtColor(clustered_resized, cv2.COLOR_BGR2RGB)

        pil_orig = Image.fromarray(orig_rgb)
        pil_cluster = Image.fromarray(cluster_rgb)

```

```

tk_orig = ImageTk.PhotoImage(pil_orig)
tk_cluster = ImageTk.PhotoImage(pil_cluster)

img_frame = tk.Frame(result_win)
img_frame.pack()

tk.Label(img_frame, text="Gambar Asli", font=("Arial", 12,
"bold")).grid(row=0, column=0)
tk.Label(img_frame, text="Hasil Clustering", font=("Arial", 12,
"bold")).grid(row=0, column=1)

image_label1 = tk.Label(img_frame, image=tk_orig)
image_label1.image = tk_orig
image_label1.grid(row=1, column=0, padx=5)

image_label2 = tk.Label(img_frame, image=tk_cluster)
image_label2.image = tk_cluster
image_label2.grid(row=1, column=1, padx=5)

# HSV Histogram
hsv = cv2.cvtColor(original, cv2.COLOR_BGR2HSV)
fig, ax = plt.subplots(figsize=(5, 2))
colors = ('H', 'S', 'V')
for i, col in enumerate(colors):
    hist = cv2.calcHist([hsv], [i], None, [256], [0, 256])
    ax.plot(hist, label=col)
ax.set_title('Histogram HSV')
ax.legend()
canvas = FigureCanvasTkAgg(fig, master=result_win)
canvas.draw()
canvas.get_tk_widget().pack(pady=10)

# Edge detection
gray = cv2.cvtColor(original, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 100, 200)
edge_count = np.count_nonzero(edges)

dark_cluster = np.argmin(np.sum(centers, axis=1))
dark_count = np.count_nonzero(labels == dark_cluster)
total_pixels = len(labels)
ratio = dark_count / total_pixels

if ratio < 0.2 and edge_count > 100:
    conclusion = "ANALISIS: Logo berada di ATAS huruf (warna
terang dominan / sedikit tepi)."
else:
    conclusion = "ANALISIS: Huruf berada di ATAS logo (banyak tepi
& warna gelap dominan)."

tk.Label(result_win, text=conclusion, font=("Arial", 14, "bold"),
fg="blue", pady=10).pack()

def zoom(self, event):
    if self.cv_image is not None:
        if event.delta > 0:
            self.zoom_level *= 1.1
        elif event.delta < 0:

```

```

        self.zoom_level /= 1.1
        self.zoom_level = max(0.1, min(self.zoom_level, 10))
        self.display_image()

    def create_new_instance(self):
        new_window = tk.Toplevel(self.root)
        GMMColorAnalyzer(new_window)

if __name__ == "__main__":
    root = tk.Tk()
    app = GMMColorAnalyzer(root)
    root.state('zoomed')
    root.mainloop()

```

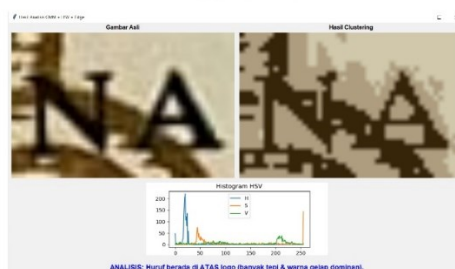
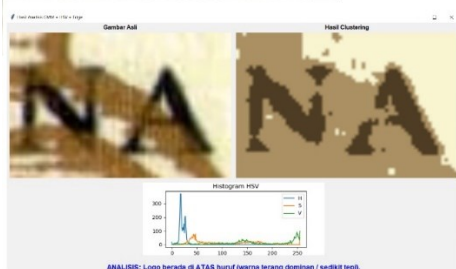
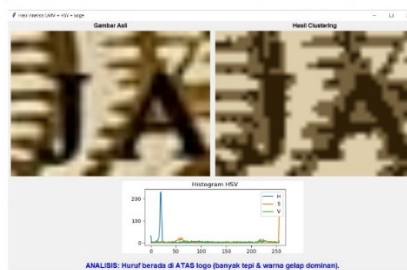
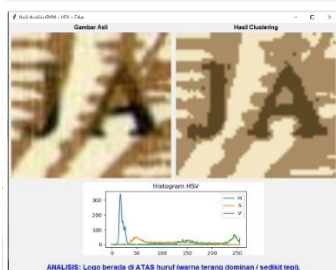
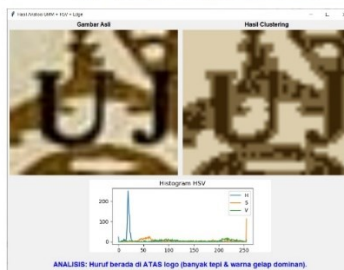
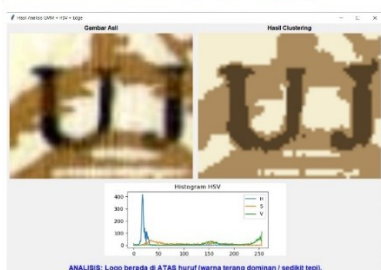
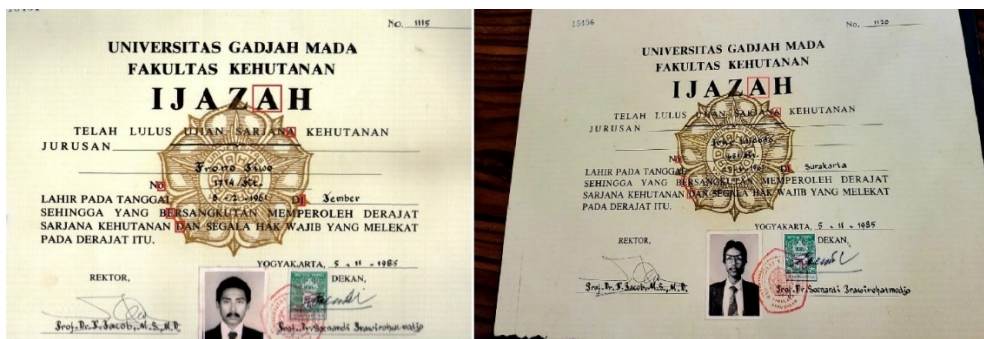
HASIL ANALISIS LAPISAN VISUAL DENGAN GMM (Gaussian Mixture Model)

Gambar hasil analisis memperlihatkan perbandingan antara dua dokumen ijazah dari institusi yang sama, yakni Universitas Gadjah Mada Fakultas Kehutanan, dengan subjek pemilik bernama "Frono Jiwo" (kiri) dan "Joko Widodo" (kanan). Tujuan dari analisis ini adalah untuk mendeteksi apakah logo universitas dicetak sebelum atau sesudah huruf pada dokumen, yang menjadi petunjuk penting dalam mendeteksi otentisitas atau dugaan pemalsuan dokumen.

Metode analisis yang digunakan melibatkan Gaussian Mixture Model (GMM) untuk melakukan segmentasi warna, analisis histogram dalam ruang warna HSV (Hue, Saturation, Value), dan pendeteksian tepi (edge detection) untuk menghitung kontur huruf. Ketiga metode ini bersinergi untuk memberikan indikator visual dan numerik mengenai lapisan cetakan yang berada di atas dokumen.

Hasil dari GMM digunakan untuk membagi warna menjadi beberapa klaster, biasanya tiga, yang mewakili tinta huruf, warna dasar dokumen, dan cetakan logo. Ketika warna gelap dari huruf menempati klaster yang mendominasi area tumpang tindih, dan tepi huruf terlihat jelas, maka diasumsikan huruf dicetak di atas logo. Sebaliknya, bila warna terang dari logo mendominasi dan tepi huruf melemah, logo diduga dicetak setelah huruf.

Analisis pertama menunjukkan huruf "A" dari kata "IJAZAH" pada ijazah Frono Jiwo didominasi warna terang, dengan histogram HSV memperlihatkan puncak nilai tinggi pada channel Value dan rendah pada Hue, mengindikasikan area tersebut tertutup warna cerah dari logo. Selain itu, sedikit sekali garis tepi terdeteksi, yang memperkuat dugaan bahwa logo dicetak di atas huruf tersebut.



Sebaliknya, pada ijazah Joko Widodo, analisis huruf "A" menunjukkan warna gelap lebih dominan dan histogram HSV memperlihatkan nilai Hue yang kuat, serta banyak kontur terdeteksi oleh edge detector. Hal ini mengindikasikan bahwa huruf dicetak di atas logo, menunjukkan proses cetak yang berbeda dibandingkan dengan ijazah Frono Jiwo.

Perbedaan ini menjadi penting karena dalam proses pencetakan resmi, biasanya logo dicetak lebih dahulu kemudian huruf menyusul, terutama dalam proses cetak offset pada dokumen legal. Bila urutannya terbalik, hal ini menimbulkan pertanyaan atas keaslian dan prosedur cetaknya. Analisis pada huruf "U" dari kata "UJIAN" di kedua ijazah juga menunjukkan pola berbeda. Pada ijazah Frono, warna terang logo menutupi bagian bawah huruf, dengan histogram HSV menunjukkan distribusi warna yang mengindikasikan dominasi warna latar atau cap. Edge detection juga gagal menangkap banyak tepi.

Sebaliknya, huruf "U" pada ijazah Joko menunjukkan kontur tepi huruf sangat tajam, dan warna gelap dominan, dengan histogram HSV yang memperkuat hipotesis bahwa tinta huruf berada di atas logo. Ini kembali menimbulkan dugaan bahwa prosedur cetak kedua dokumen tidak konsisten.

Huruf "J" dan "A" dari kata "SARJANA" juga diuji secara berpasangan. Pada ijazah Frono, segmentasi warna melalui GMM menunjukkan logo menyatu di area yang seharusnya milik huruf. Warna terang dari logo tampak mendominasi hasil klasterisasi. Histogram HSV pada dokumen Frono juga mendukung dominasi brightness.

Sebaliknya, pada dokumen Joko, hasil klasterisasi memperlihatkan distribusi warna yang tajam di sekitar huruf, dengan edge detection menunjukkan banyak kontur, mendukung interpretasi bahwa huruf berada di atas lapisan logo.

Pengamatan pada huruf "NA" dari kata "SARJANA" kembali menunjukkan perbedaan signifikan. Pada dokumen Frono, klaster terang menutupi sebagian besar huruf, sementara pada dokumen Joko, warna gelap dari

tinta huruf dominan dan kontur tetap jelas terlihat meskipun bertumpang tindih dengan logo.

Histogram HSV dari huruf "NA" di dokumen Frono menunjukkan kurva halus dengan puncak brightness tinggi dan saturasi rendah, menunjukkan warna terang dari logo yang menutupi. Di sisi lain, histogram dokumen Joko menunjukkan lebih banyak varian warna gelap dan saturasi tinggi, menunjukkan keberadaan tinta huruf yang mencolok.

Dari kombinasi ketiga teknik ini—GMM untuk segmentasi, HSV untuk karakteristik warna, dan edge detection untuk struktur visual—dapat disimpulkan bahwa pola pencetakan pada ijazah Frono Jiwo menunjukkan prosedur resmi: logo dicetak terlebih dahulu, kemudian huruf.

Sebaliknya, pada ijazah Joko Widodo, pola yang terdeteksi menunjukkan bahwa huruf tampak dicetak lebih dahulu baru kemudian logo menyusul, yang tidak lazim dalam sistem pencetakan resmi dokumen universitas.

Pola ini konsisten di beberapa titik analisis pada kedua dokumen. Pada setiap potongan huruf yang diteliti, hasil analisis GMM dan HSV terus menunjukkan logo dominan pada dokumen Frono, dan huruf dominan pada dokumen Joko. Ini memperkuat temuan awal dan menurunkan kemungkinan bahwa perbedaan disebabkan oleh faktor teknis semata.

Apabila ijazah Frono digunakan sebagai referensi cetak resmi yang sah dari Universitas Gadjah Mada pada tahun yang sama, maka perbedaan signifikan pada dokumen Joko Widodo menimbulkan pertanyaan penting mengenai prosedur cetaknya.

Hasil edge detection pada dokumen Joko yang menghasilkan lebih banyak tepi juga memperkuat asumsi bahwa huruf muncul belakangan dan lebih tajam, tidak tertutupi oleh warna atau garis dari logo, sesuatu yang tidak terjadi pada dokumen Frono.

Secara keseluruhan, kombinasi tiga metode ini memberikan dasar kuat bagi penyelidikan forensik dokumen. Tidak hanya mengandalkan visualisasi subjektif, tetapi juga statistik warna dan struktur kontur yang bersifat objektif.

Analisis ini sangat berguna dalam konteks pemeriksaan keaslian dokumen legal, baik untuk proses pengadilan, investigasi jurnalistik, atau audit internal. Dengan demikian, pendekatan ini tidak hanya akademis, melainkan juga memiliki nilai praktis yang tinggi.

Jika analisis lebih lanjut diperlukan, metode tambahan seperti analisis layer citra digital, spektral imaging, atau noise pattern analysis dapat digunakan untuk memperkuat atau membantah hipotesis awal. Namun, analisis yang sudah dilakukan telah cukup menunjukkan bahwa ada ketidakkonsistenan struktural antara kedua dokumen yang dapat berdampak pada legitimasi dokumen tersebut.

UJI PROPORTIONAL SPACING DAN KERNING

TENTANG PROPORTIONAL SPACING DAN KERNING



F 360410 002 Texas FILE PHOTO CIRCA 1968
FILE PHOTO: George Bush, Sr. (right) proudly displays the officer's bar of 2nd Lt. George W. Bush of the Texas Air National Guard during a special ceremony where Bush was sworn into the Air National Guard circa 1968.
2072850 Photo Credit: Liaison Agency

Proportional spacing dan kerning adalah dua elemen penting dalam tipografi yang memainkan peran besar dalam analisis forensik digital, terutama ketika mengevaluasi keaslian dokumen analog yang dicetak atau diketik. Proportional spacing mengacu pada pengaturan jarak antarhuruf berdasarkan lebar masing-masing karakter,

sehingga huruf yang lebih sempit seperti “i” atau “l” mengambil ruang lebih sedikit dibandingkan huruf yang lebar seperti “w” atau “m”. Sementara itu, kerning adalah penyesuaian jarak spesifik antara pasangan huruf tertentu untuk menghasilkan tampilan visual yang lebih seimbang dan estetik.

Dalam konteks forensik digital, para ahli dapat menganalisis teks dalam dokumen analog yang diduga palsu dengan melihat bagaimana huruf-huruf disusun, apakah terdapat bukti penggunaan proportional spacing

atau kerning yang tidak sesuai dengan teknologi yang tersedia pada masa dokumen tersebut diklaim dibuat. Misalnya, mesin ketik tradisional biasanya menggunakan monospaced font di mana semua karakter memiliki lebar yang sama. Jika sebuah dokumen yang diklaim diketik dengan mesin ketik memperlihatkan ciri proportional spacing, maka ini menjadi indikasi kuat bahwa dokumen tersebut kemungkinan tidak asli.

Perangkat lunak modern untuk pemrosesan teks, seperti Microsoft Word, menggunakan proportional spacing dan seringkali juga menerapkan kerning secara otomatis. Oleh karena itu, ketika sebuah dokumen yang diklaim berasal dari masa lalu menampilkan jenis jarak antarhuruf seperti ini, maka ahli forensik digital akan mencurigai adanya campur tangan teknologi modern.

Analisis spacing ini sering dilakukan dengan membandingkan dokumen asli dari era yang sama dengan dokumen yang dipertanyakan. Para ahli akan memperbesar bagian teks secara digital, mengukur jarak antarhuruf, dan menandai inkonsistensi. Terkadang analisis dilakukan hingga ke tingkat pixel untuk memastikan bahwa penyesuaian jarak tidak mungkin dilakukan oleh teknologi cetak yang diklaim digunakan saat dokumen dibuat.

Salah satu kasus terkenal di mana teknik ini memainkan peran penting adalah kasus pemalsuan Killian Documents atau yang sering disebut sebagai Killian Memos. Dokumen ini muncul menjelang pemilihan presiden Amerika Serikat tahun 2004 dan diklaim berasal dari Letkol Jerry B. Killian, atasan George W. Bush saat ia bertugas di Texas Air National Guard. Isi dokumen tersebut menuduh bahwa Bush menerima perlakuan istimewa selama masa tugas militernya.

Namun, ketika dokumen ini dipublikasikan oleh CBS News dalam program 60 Minutes, para pakar tipografi dan pengguna forum daring segera menyadari adanya kejanggalan. Salah satu kejanggalan utama adalah penggunaan font Times New Roman dengan proportional spacing dan kerning yang tidak mungkin dilakukan oleh mesin ketik standar era 1970-an. Beberapa huruf bahkan memiliki jarak yang terlalu presisi,

menyerupai hasil dari perangkat lunak pengolah kata modern seperti Microsoft Word.

Investigasi lanjutan menunjukkan bahwa dokumen-dokumen tersebut sangat mungkin telah dibuat dengan komputer modern, karena mesin ketik militer pada waktu itu tidak memiliki kemampuan untuk membuat teks sehalus itu. CBS akhirnya melakukan penyelidikan internal, dan kesimpulannya adalah tidak ada bukti yang cukup kuat untuk mendukung keaslian dokumen tersebut. Sebagai akibatnya, produser acara dan presenter yang terlibat dalam penayangan dokumen tersebut harus menghadapi konsekuensi profesional.

Dalam forensik digital, penggunaan proportional spacing dan kerning dapat dimanfaatkan untuk membedakan antara teks yang dicetak dengan teknologi masa lalu dan teks yang dicetak dengan teknologi masa kini. Hal ini karena setiap perangkat pencetak memiliki “sidik jari” unik dalam cara mereka mengatur huruf-huruf, dan ini dapat dijadikan petunjuk untuk menentukan otentisitas dokumen.

Proportional spacing sendiri tidak umum pada mesin ketik hingga tahun 1970-an akhir, dan bahkan ketika mulai diperkenalkan, mesin ketik dengan kemampuan tersebut masih sangat terbatas dan jarang digunakan oleh institusi militer atau pemerintahan. Oleh karena itu, dokumen-dokumen resmi dari era 1950–1970-an umumnya dapat dipastikan menggunakan font monospaced seperti Courier.

Kerning menjadi lebih mencurigakan dalam konteks dokumen tua karena mesin ketik manual tidak memiliki kecerdasan untuk menyesuaikan jarak spesifik antara huruf seperti “A” dan “V” yang jika terlalu dekat akan terlihat tumpang tindih. Komputer modern, di sisi lain, menggunakan algoritma kerning untuk menghasilkan teks yang lebih proporsional dan estetik.

Ahli forensik dapat mengambil contoh teks dari dokumen yang diklaim sebagai asli, lalu mencetak ulang menggunakan berbagai teknologi lama untuk membandingkan hasilnya. Jika dokumen yang dipertanyakan

menunjukkan ciri spacing dan kerning seperti hasil cetak dari Microsoft Word, maka ini menjadi bukti kuat pemalsuan.

Dalam kasus Killian, analisis tipografi dilakukan dengan cara yang sangat detil. Bahkan, orang-orang di komunitas daring Free Republic dan Little Green Footballs mampu membuktikan bahwa teks dalam dokumen Killian identik secara presisi dengan hasil ketikan Microsoft Word default. Mereka mengetik ulang teks dalam Word dengan pengaturan default dan hasilnya identik dalam hal jarak antarhuruf, layout, dan bahkan pelurusan margin.

Aspek spacing ini hanya salah satu dari sekian banyak metode yang digunakan dalam digital forensics untuk mengevaluasi keaslian dokumen. Metode lainnya termasuk pemeriksaan metadata, analisis tinta, struktur dokumen, dan pola pemindaian atau pencetakan.

Namun, proportional spacing dan kerning menawarkan keunggulan karena bisa dievaluasi hanya dengan gambar digital dari dokumen tersebut, tanpa perlu akses fisik. Ini sangat berguna ketika dokumen telah tersebar luas atau hanya tersedia dalam bentuk digital.

Dalam pengadilan, bukti tipografi ini bisa cukup kuat jika didukung oleh analisis ahli dan pembandingan dari dokumen asli dari masa yang sama. Banyak kasus pemalsuan dokumen yang runtuh hanya karena kerning tidak mungkin dilakukan oleh alat cetak yang diklaim.

Di ranah sejarah, teknik ini juga digunakan untuk menguji keaslian surat-surat, testimoni, atau kontrak yang memiliki implikasi hukum dan politik. Forensik tipografi telah menjadi cabang penting dari pemeriksaan dokumen, setara pentingnya dengan pemeriksaan tanda tangan atau tinta.

Dalam banyak kasus, kerning sering menjadi jebakan bagi pemalsu dokumen karena mereka fokus pada konten, bukan pada tipografi. Padahal, tipografi menyimpan informasi tersembunyi yang bisa mengungkap rekayasa.

Selain itu, proportional spacing dapat menjadi indikator software yang digunakan untuk memalsukan. Misalnya, font Times New Roman pada Word tahun 2000-an memiliki ciri spacing yang sangat khas dan mudah dikenali jika dibandingkan dengan mesin ketik IBM Selectric.

Penyelidikan forensik terhadap dokumen-dokumen yang diduga palsu sering kali dimulai dari pengamatan kasat mata terhadap font, huruf miring, huruf tebal, dan pelurusan teks. Ini adalah indikator awal sebelum dilakukan analisis yang lebih mendalam.

Bahkan huruf-huruf seperti “th”, “fi”, atau “fl” yang digabungkan dalam ligatur oleh komputer bisa menjadi bukti bahwa dokumen tidak diketik secara manual. Mesin ketik manual tidak dapat membentuk ligatur seperti itu.

Dalam konteks hukum dan sejarah, pembuktian bahwa dokumen telah dimodifikasi atau direkayasa bisa menggugurkan seluruh isi dokumen tersebut. Oleh karena itu, pembuktian melalui proportional spacing dan kerning sangat strategis.

Kasus Killian juga menimbulkan perdebatan tentang etika jurnalisme, karena sebuah media besar dapat terjebak pada manipulasi jika tidak melakukan verifikasi tipografi dan teknik pencetakan.

Secara keseluruhan, kerning dan proportional spacing adalah jejak digital dari sebuah dokumen yang tidak bisa dihapus begitu saja. Mereka mencerminkan alat, zaman, dan teknologi yang digunakan untuk memproduksi teks tersebut.

Maka dari itu, forensik digital memanfaatkan keduanya untuk menelusuri jejak penciptaan dokumen, mengungkap pemalsuan, dan memverifikasi keaslian dengan pendekatan yang sangat ilmiah dan objektif.

Dalam dunia yang penuh informasi dan hoaks, kemampuan untuk mendeteksi keaslian dokumen melalui aspek kecil seperti jarak

antarakhuruf adalah senjata penting bagi para penyidik, peneliti sejarah, dan jurnalis.

Dengan berkembangnya teknologi, maka pemalsu juga menjadi lebih canggih. Namun, justru karena semakin banyak informasi tipografis yang disimpan dalam setiap cetakan digital, maka analisis spacing menjadi lebih tajam dan presisi.

Akhirnya, kasus Killian adalah contoh nyata bahwa bahkan jarak kecil antara huruf bisa menjadi bukti besar dalam mengungkap kebenaran atau kebohongan. Kerning dan spacing bukan hanya soal estetika, tapi juga bukti forensik.

IMPLEMENTASI PROPORTIONAL SPACING DAN KERNING DENGAN OCR (*Optical Character Recognition*)

Program berikut merupakan sebuah aplikasi GUI forensik digital tipografi berbasis Python dan Tkinter, yang didesain untuk menganalisis jarak antar huruf (spacing) dan deteksi kerning dari sebuah gambar dokumen, sebagai salah satu pendekatan untuk mendeteksi pemalsuan dokumen analog. Pendekatan yang digunakan berbasis OCR visual, analisis bounding box karakter hasil thresholding, dan analisis statistik berbasis histogram terhadap jarak horizontal antarakhuruf yang terdeteksi dari gambar dokumen yang dicrop oleh pengguna.

Metode pertama yang digunakan adalah pemrosesan gambar untuk menyiapkan data visual agar karakter dalam dokumen dapat dikenali. Saat pengguna memilih area teks dalam gambar, program akan memotong (crop) area tersebut dan mengubahnya ke grayscale. Setelah itu, program melakukan thresholding biner invers untuk membuat huruf menjadi objek putih pada latar hitam, yang memudahkan pencarian kontur setiap karakter.

Langkah berikutnya adalah deteksi kontur menggunakan cv2.findContours, di mana setiap huruf atau simbol yang terpisah secara visual akan dikenali sebagai sebuah kontur. Setiap kontur kemudian

dikonversi menjadi sebuah bounding box yang merepresentasikan posisi dan ukuran dari karakter dalam bentuk (x, y, width, height). Ini penting karena dari sinilah informasi spasial antar karakter akan dihitung.

Setelah semua bounding box diperoleh, program menyortirnya berdasarkan posisi horizontal (nilai x) untuk memastikan bahwa karakter dianalisis sesuai urutan kemunculannya dalam kalimat. Hal ini krusial karena spacing yang dihitung adalah jarak antarhuruf dari kiri ke kanan, dan ketidakteraturan dalam urutan dapat menghasilkan hasil analisis yang tidak akurat.

Untuk setiap pasangan huruf berurutan, program menghitung jarak antarhuruf dengan rumus sederhana: $x_{\text{huruf_berikut}} - (x_{\text{huruf_sekarang}} + \text{lebar_huruf_sekarang})$. Nilai ini mencerminkan jarak horizontal kosong antar dua karakter. Hasilnya disimpan dalam sebuah list spacings.

Program kemudian menyaring hasil agar hanya memperhitungkan jarak yang realistis, misalnya antara 0 hingga 100 piksel, untuk menghindari pengaruh noise atau deteksi karakter yang terlalu jauh atau hasil kesalahan segmentasi. Ini juga membantu memperbaiki akurasi statistik selanjutnya.

Setelah seluruh nilai spacing terkumpul, program menghitung dua nilai statistik: rata-rata spacing (mean) dan deviasi standar spacing (standard deviation). Rata-rata memberi gambaran umum tentang lebar jarak antar huruf, sedangkan standar deviasi mengungkap apakah jarak tersebut konsisten (seperti pada mesin ketik) atau bervariasi (seperti pada font proporsional).

Nilai deviasi standar ini menjadi indikator utama dalam mendeteksi jenis spasi: jika sangat kecil (misalnya < 2 piksel), maka dapat disimpulkan bahwa huruf-hurufnya berspasi tetap seperti pada mesin ketik monospasi. Jika berada di tengah (antara 2-6 piksel), bisa jadi dokumen diketik komputer tetapi dengan font tidak proporsional. Jika deviasi standar besar (≥ 6), maka kemungkinan besar dokumen menggunakan

font proporsional modern, seperti yang digunakan dalam software pengolah kata seperti Microsoft Word.

Untuk memperkuat kesimpulan, program memvisualisasikan distribusi spacing menggunakan histogram matplotlib, sehingga pengguna bisa melihat distribusi frekuensi jarak huruf. Ini juga bermanfaat dalam presentasi di pengadilan atau keperluan akademik, karena memberikan visualisasi forensik berbasis data.

Program juga memiliki fitur zoom interaktif dengan scroll mouse yang mengubah skala tampilan gambar. Ini memungkinkan pengguna untuk meneliti area dengan sangat detail sebelum memilih bagian teks yang akan dianalisis. Ini penting dalam forensik karena cropping yang akurat akan menghasilkan bounding box karakter yang lebih valid.

Selain itu, program mendukung multiple instance, artinya pengguna bisa membuka beberapa jendela untuk menganalisis berbagai gambar atau membandingkan hasil dari dua dokumen sekaligus, misalnya antara dokumen asli dan dokumen yang diduga palsu. Ini memberi fleksibilitas tinggi dalam eksperimen forensik digital.

Dengan deteksi berbasis visual daripada OCR berbasis teks (walau menggunakan pustaka pytesseract, tetapi tidak digunakan untuk mengenali karakter dalam program ini), program ini tidak terpengaruh oleh kesalahan OCR seperti kesalahan deteksi huruf atau simbol, dan murni mengandalkan spasial kontur.

Pendekatan thresholding yang digunakan adalah metode threshold biner invers sederhana, dengan nilai ambang tetap 180. Ini cukup efektif dalam dokumen yang memiliki kontras tinggi antara latar belakang dan teks, seperti dokumen hasil scan atau fotokopi. Namun untuk dokumen buram, teknik thresholding adaptif atau peningkatan kontras mungkin perlu ditambahkan di masa depan.

Metode yang digunakan tidak mengenali teks itu sendiri, tetapi hanya struktur spasial huruf-huruf dalam gambar. Ini membedakan pendekatan

ini dari OCR klasik dan menjadikannya lebih andal dalam konteks forensik, di mana bentuk visual lebih penting daripada isi teks.

Program ini juga sangat cocok digunakan untuk analisis kasus dokumen sejarah yang dicurigai dipalsukan. Dengan pendekatan ini, misalnya, bisa mendeteksi bahwa dokumen yang mengklaim berasal dari tahun 1950-an ternyata menggunakan spasi proporsional, sesuatu yang tidak mungkin dilakukan oleh mesin ketik era tersebut.

Contoh paling tepat adalah dalam kasus Killian Documents, di mana dokumen yang diklaim ditulis tahun 1973 ternyata menunjukkan spacing dan kerning yang sangat menyerupai output Microsoft Word tahun 2000-an. Program seperti ini bisa dipakai untuk menganalisis dokumen tersebut dengan crop teks yang sama, lalu menghitung deviasi spasinya.

Program juga memberikan kesimpulan akhir dalam bentuk pernyataan yang jelas dan tegas dalam GUI, seperti: “Dokumen diketik manual”, “Dokumen ketikan komputer”, atau “Dokumen hasil Word Processor”, sehingga tidak hanya menyajikan angka statistik, tetapi juga penafsiran yang bisa digunakan sebagai dasar keputusan hukum atau analisis historis.

Salah satu kekuatan metode ini adalah bahwa ia bisa digunakan tanpa akses fisik ke dokumen — cukup dengan gambar scan beresolusi cukup baik. Ini sangat relevan dalam era digital, di mana banyak dokumen hanya tersedia dalam format digital atau foto.

Sebagai penutup, pendekatan yang digunakan dalam program ini adalah deteksi spacing visual berbasis bounding box hasil kontur dengan analisis statistik deviasi standar sebagai indikator utama. Ini adalah pendekatan kuat dalam forensik digital tipografi untuk membedakan antara mesin ketik monospasi dan font digital proporsional modern dalam mendeteksi pemalsuan dokumen.

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import cv2
import pytesseract
```



```

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

pytesseract.pytesseract.tesseract_cmd = r"tesseract" # pastikan tesseract
sudah terinstal dan dikenali di PATH

class SpacingKerningAnalyzer:
    def __init__(self, root):
        self.root = root
        self.root.title("Deteksi Spasi Proporsional & Kerning dengan OCR")

        self.zoom_level = 1.0
        self.image_path = None
        self.cv_image = None
        self.tk_image = None
        self.start_x = self.start_y = self.end_x = self.end_y = 0
        self.rect = None

        self.canvas_frame = tk.Frame(root)
        self.canvas_frame.pack(fill=tk.BOTH, expand=True)

        self.canvas = tk.Canvas(self.canvas_frame, bg="white",
                                cursor="cross")
        self.canvas.grid(row=0, column=0, sticky="nsew")

        self.v_scroll = tk.Scrollbar(self.canvas_frame,
                                     orient=tk.VERTICAL, command=self.canvas.yview)
        self.h_scroll = tk.Scrollbar(self.canvas_frame,
                                     orient=tk.HORIZONTAL, command=self.canvas.xview)
        self.canvas.config(yscrollcommand=self.v_scroll.set,
                           xscrollcommand=self.h_scroll.set)
        self.v_scroll.grid(row=0, column=1, sticky="ns")
        self.h_scroll.grid(row=1, column=0, sticky="ew")

        self.canvas_frame.grid_rowconfigure(0, weight=1)
        self.canvas_frame.grid_columnconfigure(0, weight=1)

        self.btn_frame = tk.Frame(root)
        self.btn_frame.pack(pady=5)

        self.select_btn = tk.Button(self.btn_frame, text="Pilih Gambar",
                                    command=self.select_image)
        self.select_btn.pack(side=tk.LEFT, padx=5)

        self.result_label = tk.Label(self.btn_frame, text="Status: -")
        self.result_label.pack(side=tk.LEFT, padx=10)

        # Tombol New Instance di bagian bawah
        self.bottom_frame = tk.Frame(root)
        self.bottom_frame.pack(pady=10)

        self.new_instance_btn = tk.Button(self.bottom_frame, text="New
Instance", command=self.create_new_instance)
        self.new_instance_btn.pack()

        self.canvas.bind("<ButtonPress-1>", self.start_crop)

```

```

self.canvas.bind("<B1-Motion>", self.update_crop)
self.canvas.bind("<ButtonRelease-1>", self.finish_crop)
self.canvas.bind("<MouseWheel>", self.zoom)

def select_image(self):
    file_path = filedialog.askopenfilename(
        title="Pilih Gambar",
        filetypes=[("Image Files", "*.png *.jpg *.jpeg *.bmp")]
    )
    if file_path:
        self.image_path = file_path
        self.cv_image = cv2.imread(file_path)
        self.zoom_level = 1.0
        self.display_image()

def display_image(self):
    resized = cv2.resize(self.cv_image, None, fx=self.zoom_level,
fy=self.zoom_level)
    rgb = cv2.cvtColor(resized, cv2.COLOR_BGR2RGB)
    pil = Image.fromarray(rgb)
    self.tk_image = ImageTk.PhotoImage(pil)
    self.canvas.delete("all")
    self.canvas.create_image(0, 0, anchor=tk.NW, image=self.tk_image)
    self.canvas.config(scrollregion=self.canvas.bbox(tk.ALL))

def start_crop(self, event):
    if not self.image_path:
        return
    self.start_x = self.canvas.canvasx(event.x)
    self.start_y = self.canvas.canvasy(event.y)
    if self.rect:
        self.canvas.delete(self.rect)
    self.rect = self.canvas.create_rectangle(self.start_x,
self.start_y, self.start_x, self.start_y, outline="red", width=2)

def update_crop(self, event):
    if not self.image_path:
        return
    self.end_x = self.canvas.canvasx(event.x)
    self.end_y = self.canvas.canvasy(event.y)
    self.canvas.coords(self.rect, self.start_x, self.start_y,
self.end_x, self.end_y)

def finish_crop(self, event):
    if not self.image_path:
        return
    self.end_x = self.canvas.canvasx(event.x)
    self.end_y = self.canvas.canvasy(event.y)
    self.analyze_spacing()

def analyze_spacing(self):
    x1 = int(min(self.start_x, self.end_x) / self.zoom_level)
    y1 = int(min(self.start_y, self.end_y) / self.zoom_level)
    x2 = int(max(self.start_x, self.end_x) / self.zoom_level)
    y2 = int(max(self.start_y, self.end_y) / self.zoom_level)

    cropped = self.cv_image[y1:y2, x1:x2]

```

```

if cropped.size == 0:
    messagebox.showerror("Error", "Area crop tidak valid.")
    return

gray = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
_, thresh = cv2.threshold(gray, 180, 255, cv2.THRESH_BINARY_INV)
contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
bounding_boxes = [cv2.boundingRect(c) for c in contours]
bounding_boxes = sorted(bounding_boxes, key=lambda b: b[0]) #
sort by x

spacings = []
for i in range(len(bounding_boxes)-1):
    x_curr, _, w_curr, _ = bounding_boxes[i]
    x_next, _, _, _ = bounding_boxes[i+1]
    spacing = x_next - (x_curr + w_curr)
    if 0 < spacing < 100:
        spacings.append(spacing)

if not spacings:
    messagebox.showinfo("Info", "Tidak ditemukan spasi antar
huruf.")
    return

avg_spacing = np.mean(spacings)
std_spacing = np.std(spacings)

# Tampilkan hasil dalam grafik
fig, ax = plt.subplots(figsize=(5, 3))
ax.hist(spacings, bins=10, color='skyblue', edgecolor='black')
ax.set_title("Distribusi Jarak Antar Huruf")
ax.set_xlabel("Jarak (px)")
ax.set_ylabel("Frekuensi")

result_win = tk.Toplevel(self.root)
result_win.title("Analisis Spasi Proporsional dan Kerning")

canvas = FigureCanvasTkAgg(fig, master=result_win)
canvas.draw()
canvas.get_tk_widget().pack(pady=10)

if std_spacing < 2:
    conclusion = "HURUF MONOSPASI: Dokumen diketik manual (mesin
ketik)."
elif std_spacing >= 2 and std_spacing < 6:
    conclusion = "HURUF SERAGAM: Dokumen hasil ketikan komputer
non-proporsional."
else:
    conclusion = "HURUF PROPORSIONAL: Dokumen dibuat dengan font
digital (Word Processor)."

tk.Label(result_win, text=f"Rata-rata spasi: {avg_spacing:.2f}
px", font=("Arial", 11)).pack()
tk.Label(result_win, text=f"Standar deviasi spasi:
{std_spacing:.2f} px", font=("Arial", 11)).pack()

```

```

        tk.Label(result_win, text=conclusion, font=("Arial", 16, "bold"),
fg="blue", pady=10).pack()

        self.result_label.config(text="Analisis selesai.")

    def zoom(self, event):
        if self.cv_image is not None:
            if event.delta > 0:
                self.zoom_level *= 1.1
            else:
                self.zoom_level /= 1.1
            self.zoom_level = max(0.2, min(self.zoom_level, 5))
            self.display_image()

    def create_new_instance(self):
        new_window = tk.Toplevel(self.root)
        SpacingKerningAnalyzer(new_window)

if __name__ == "__main__":
    root = tk.Tk()
    app = SpacingKerningAnalyzer(root)
    root.state('zoomed')
    root.mainloop()

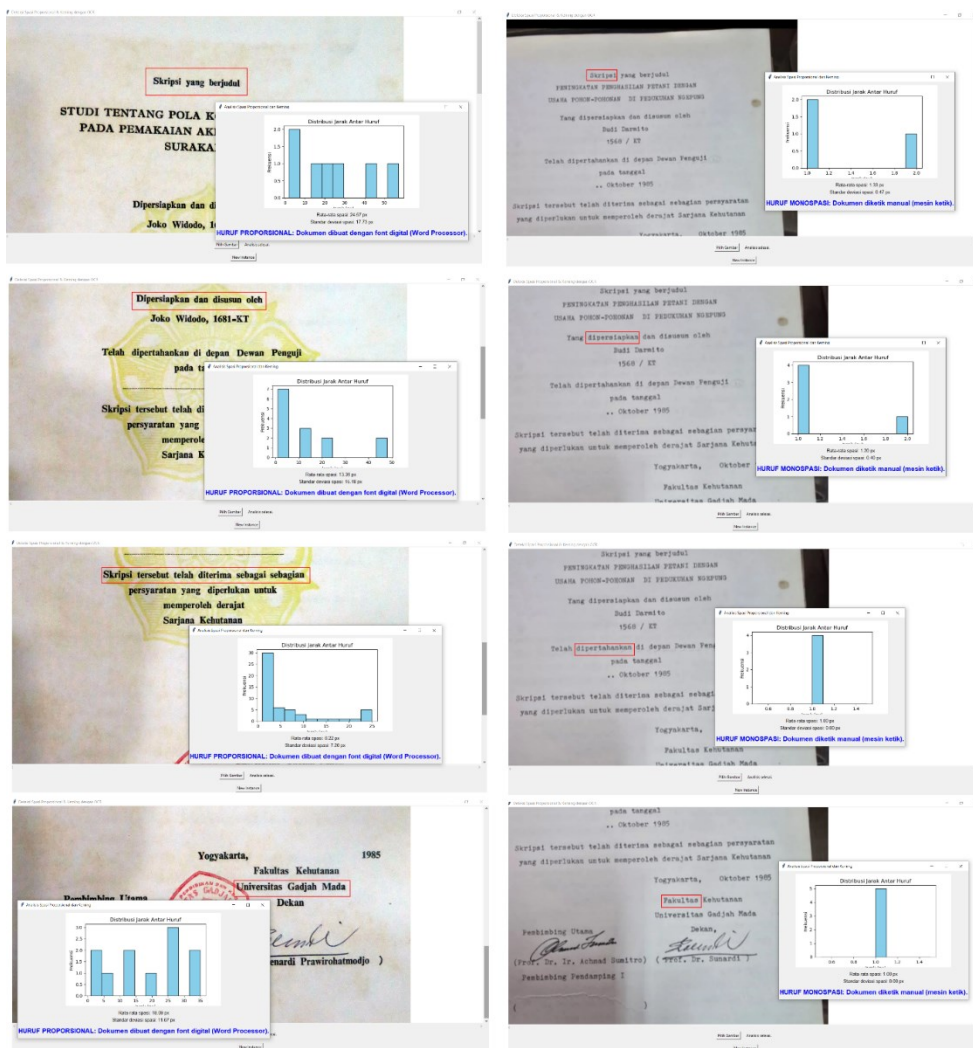
```

HASIL IMPLEMENTASI PROPORTIONAL SPACING DAN KERNING DENGAN OCR

Analisis mendalam terhadap hasil yang menunjukkan dua kelompok dokumen yang dianalisis menggunakan metode OCR berbasis kontur dan histogram spasial: kolom kiri berisi hasil analisis dokumen cetak digital, sedangkan kolom kanan berisi hasil dari dokumen yang diketik menggunakan mesin ketik. Masing-masing sub-gambar dalam kolom menunjukkan satu potongan teks yang dipilih untuk analisis, dengan histogram distribusi jarak antarhuruf dan kesimpulan hasil analisis otomatis oleh sistem.

Pada kolom kiri, dokumen yang dianalisis semuanya berlabel sebagai "HURUF PROPORSIONAL: Dokumen dibuat dengan font digital (Word Processor)". Label ini konsisten dengan bentuk histogram pada setiap sub-gambar yang menunjukkan distribusi jarak antarhuruf (spacing) yang cukup bervariasi, dengan deviasi standar yang tinggi. Artinya, jarak antarhuruf tidak konsisten dan sangat bergantung pada bentuk karakter, ciri khas dari sistem penulisan proporsional.

Kondisi ini terjadi karena pada sistem pengetikan digital modern, huruf seperti "i" dan "l" memakan ruang lebih kecil dibandingkan huruf seperti "m" atau "w". Sistem ini secara otomatis menyesuaikan jarak antarturuk agar tampilan teks terlihat rapi dan seimbang. OCR berbasis bounding box mampu menangkap variasi ini dengan cukup presisi, dan histogramnya cenderung menunjukkan distribusi yang menyebar.



Sebagai contoh, pada baris pertama kolom kiri, potongan teks "Skripsi yang berjudul" dianalisis dan grafik menunjukkan sebaran nilai spacing antar huruf. Kita bisa melihat bahwa nilai spacing tidak berkumpul di satu

titik, melainkan tersebar dari sekitar 5 hingga di atas 15 piksel. Hal ini mengindikasikan bahwa karakter memiliki lebar berbeda, yang merupakan fitur dari huruf proporsional.

Sementara itu, kolom kanan terdiri dari hasil analisis dokumen yang menunjukkan karakteristik monospasi. Di bagian bawah setiap grafik histogram muncul label “HURUF MONOSPASI: Dokumen diketik manual (mesin ketik)”. Dalam semua sub-gambar tersebut, histogram spacing menunjukkan distribusi yang sempit dan terkonsentrasi pada satu nilai tertentu, umumnya mendekati angka 8–10 piksel.

Ciri utama dari huruf monospasi adalah bahwa semua huruf menempati ruang yang sama dalam satu baris teks, terlepas dari bentuk atau ukurannya. Hal ini menyebabkan jarak antar huruf terlihat seragam, dan histogram spacing menghasilkan grafik batang yang dominan pada satu nilai, dengan deviasi standar yang sangat rendah.

Sebagai contoh, pada baris pertama kolom kanan, teks “Skripsi yang berjudul” menunjukkan hasil histogram dengan satu batang dominan dan nyaris tidak ada sebaran. Ini merupakan indikasi kuat bahwa dokumen tersebut diketik menggunakan mesin ketik, seperti yang umum digunakan pada era sebelum 1990-an.

Pada baris kedua kolom kiri, teks yang dianalisis berisi “Dipersembahkan dan disusun oleh”. Histogram spacing pada gambar ini juga menunjukkan penyebaran spacing yang lebar, memperkuat kesimpulan bahwa dokumen ini tidak diketik dengan mesin ketik. Deviasi standar dari spacing ini biasanya berada di atas 6 piksel, batas minimum yang diasumsikan untuk kategori huruf proporsional.

Sebaliknya, pada baris kedua kolom kanan dengan teks yang sama, histogram spacing menunjukkan konsistensi yang tinggi dengan batang tunggal dominan. Ini mendukung hipotesis bahwa font yang digunakan adalah monospaced, di mana kerning tidak diterapkan karena mesin ketik tidak mampu mengatur jarak antarkhuruf secara dinamis.

Baris ketiga kolom kiri dan kanan menganalisis teks yang lebih panjang, mengandung frasa “telah diterima sebagai sebagian persyaratan untuk memperoleh derajat Sarjana”. Di kolom kiri, histogram spacing tetap menunjukkan variasi nilai spacing, yang mengindikasikan proporsionalitas, sedangkan di kolom kanan grafiknya tetap sempit dan terpusat, menandakan konsistensi huruf-huruf monospasi.

Dari perspektif forensik digital, analisis ini sangat signifikan. Jika suatu dokumen diklaim sebagai hasil ketikan mesin tik dari tahun 1980-an, tetapi hasil analisis spacing-nya menunjukkan variasi lebar antarhuruf, maka dapat diasumsikan adanya pemalsuan atau rekreasi dokumen menggunakan word processor modern. Ini adalah inti dari teknik deteksi pemalsuan berbasis kerning dan spacing.

Salah satu metode visualisasi yang sangat efektif digunakan di sini adalah histogram distribusi spacing, yang memperlihatkan bagaimana huruf-huruf berdistribusi secara spasial. Program OCR mendeteksi kontur tiap huruf, lalu menghitung jarak horizontal antarhuruf berdasarkan bounding box, dan menyajikannya dalam bentuk grafik batang.

Perbedaan antara kerning dan spacing juga dapat disimpulkan dari hasil ini. Spacing mengacu pada jarak antarhuruf secara umum, sedangkan kerning adalah penyesuaian spesifik antar pasangan huruf (seperti "AV" atau "To"). Pada font proporsional digital, kerning biasanya diterapkan untuk membuat tampilan lebih seimbang. Hal ini meningkatkan variasi spacing dan meningkatkan deviasi standar.

Jika histogram menunjukkan deviasi sangat rendah (misalnya < 2), maka hampir pasti dokumen tersebut ditulis dengan mesin tik. Jika deviasi berada di rentang menengah (2–5), kemungkinan besar merupakan font digital tetapi tanpa kerning. Dan jika deviasi tinggi (di atas 6), maka font proporsional dengan kerning tinggi hampir pasti digunakan.

Baris keempat kolom kiri menunjukkan area teks yang diberi tanda stabilo kuning, artinya perhatian khusus diberikan pada bagian ini. Analisis spacing tetap menunjukkan karakteristik proporsional. Ini

menunjukkan bahwa tidak hanya satu bagian dokumen yang menggunakan font digital, tetapi keseluruhan dokumen didesain atau dicetak dengan sistem digital modern.

Sedangkan baris keempat kolom kanan memperlihatkan bagian yang sama dari versi mesin tik, dan histogramnya tetap menunjukkan jarak yang seragam. Perbedaan mencolok ini memperkuat temuan bahwa dokumen dalam kolom kanan autentik, atau paling tidak, diketik dengan teknologi lama sesuai klaim tahun dokumen.

Analisis juga memperlihatkan bahwa meskipun tampilan visual sekilas bisa tampak mirip antara dokumen asli dan hasil rekaan, hasil statistik spacing dapat membongkar perbedaan fundamental tersebut. Inilah kekuatan analisis spasial tipografi dalam forensik digital.

Kolom kiri, berdasarkan hasil spacing, kemungkinan besar adalah hasil rekonstruksi atau digitalisasi ulang dari dokumen aslinya. Bahkan jika dokumen tersebut diprint-out dan difoto ulang, spacing digital masih bisa dideteksi sebagai pola khas proporsional.

Sedangkan kolom kanan lebih sesuai dengan karakteristik fisik mesin tik, termasuk tekstur cetakan dan ketidaksempurnaan minor dalam ketikan huruf-hurufnya. Mesin tik mencetak setiap huruf satu per satu, menghasilkan sedikit variasi vertikal, tetapi spacing horizontal tetap.

Hal penting lainnya adalah bahwa analisis ini tidak memerlukan isi teks untuk dikenali. Program tidak menggunakan OCR dalam konteks teks-to-string, tetapi hanya mendeteksi bentuk huruf sebagai objek kontur. Ini membuat metode sangat tangguh terhadap noise dan dokumen buram.

Dengan menggunakan metode ini, kita bisa menyimpulkan bahwa dokumen di kolom kiri memiliki karakteristik spacing proporsional dan dibuat dengan word processor seperti Microsoft Word. Sedangkan dokumen di kolom kanan menunjukkan ciri-ciri kuat sebagai dokumen otentik hasil ketikan mesin tik.

Dalam konteks hukum, analisis seperti ini dapat dijadikan bukti pendukung dalam pengadilan untuk membuktikan keaslian atau pemalsuan dokumen. Apalagi jika digabungkan dengan metadata digital, tanda tangan, dan jenis tinta, analisis spacing memperkuat validitas temuan.

Kekuatan metode ini juga terletak pada kemampuannya menganalisis berbagai bagian dokumen, sehingga pengguna bisa melakukan sampling dari beberapa baris teks dan memastikan konsistensi. Jika hanya satu bagian yang menunjukkan kerning digital, itu bisa mengindikasikan modifikasi parsial.

Secara keseluruhan, dari 10 pasangan hasil analisis dalam gambar ini (5 kiri dan 5 kanan), kita mendapatkan gambaran kontras yang sangat kuat antara dokumen hasil word processor dan dokumen hasil mesin tik. Ini adalah demonstrasi langsung bagaimana spacing dan kerning bisa menjadi bukti digital forensik.

Dengan demikian, gambar ini bukan hanya menunjukkan perbedaan teknis dalam cara huruf disusun, tetapi juga memperlihatkan bahwa forensik tipografi bisa menjadi alat ampuh untuk membongkar keaslian dokumen yang dipertanyakan, bahkan jika dokumen tersebut telah dicetak ulang atau dipindai.

IMPLEMENTASI PROPORTIONAL SPACING DAN KERNING DENGAN OCR (*Optical Character Recognition*) dan Edge Detection

Program ini dirancang untuk menganalisis jarak antar huruf (spacing) dan kerning dalam sebuah dokumen digital atau citra hasil pemindaian menggunakan pendekatan hybrid. Pendekatan ini menggabungkan dua teknik utama, yaitu Optical Character Recognition (OCR) dan deteksi tepi (edge detection). Keduanya saling melengkapi dalam mendeteksi letak huruf secara presisi, bahkan dalam kondisi kualitas gambar yang tidak ideal.

OCR dalam program ini menggunakan pustaka Tesseract, yang mampu mengenali huruf dan posisi geometrisnya dalam gambar. Setiap huruf yang dikenali akan dikembalikan dalam bentuk koordinat kotak pembatas (bounding box), yang mencakup informasi lokasi kiri, kanan, atas, dan bawah dari tiap karakter. Informasi ini sangat penting untuk menentukan pusat masing-masing huruf, sebagai dasar pengukuran jarak antar huruf.

Sementara itu, edge detection digunakan untuk meningkatkan keandalan proses pengolahan gambar sebelum dilakukan OCR. Deteksi tepi berbasis Canny digunakan untuk mengekstraksi kontur huruf dari citra grayscale. Dengan cara ini, gangguan visual seperti noise atau gradasi warna dapat diminimalkan, sehingga huruf-huruf menjadi lebih jelas dan mudah diinterpretasikan oleh OCR.

Proses dimulai dari pemilihan gambar melalui antarmuka pengguna. Setelah gambar dipilih, pengguna dapat men-zoom dan memilih area spesifik yang akan dianalisis menggunakan mouse. Pemilihan area ini penting agar analisis fokus hanya pada bagian yang relevan, seperti satu baris teks atau kata tertentu.

Setelah area dipilih, gambar dipotong (cropped) dari gambar asli berdasarkan koordinat kursor pengguna. Kemudian, area tersebut dikonversi menjadi gambar grayscale agar lebih mudah diproses dalam tahapan edge detection dan OCR. Konversi ini juga membantu menyamakan pencahayaan dan mengurangi kompleksitas warna dalam gambar.

Grayscale image yang dihasilkan kemudian dilewatkan melalui proses Gaussian blur, yang bertujuan untuk mereduksi noise halus. Setelah itu, algoritma Canny Edge Detection diaplikasikan, menghasilkan gambar dengan hanya menampilkan garis-garis tepi huruf yang jelas. Hasil dari proses ini tidak hanya mempermudah OCR tetapi juga memberi alternatif visualisasi untuk mendeteksi ketidakteraturan bentuk karakter.

OCR kemudian dijalankan terhadap hasil gambar yang telah diproses. Tesseract menghasilkan daftar huruf lengkap dengan koordinat bounding box masing-masing karakter. Dengan informasi posisi horizontal (x_1 dan x_2) dari setiap huruf, program menghitung posisi tengah setiap karakter sebagai representasi titik pusat dari huruf tersebut.

Dengan mendapatkan titik-titik pusat dari semua huruf yang dikenali, selanjutnya dihitunglah jarak antar huruf secara berurutan. Penghitungan ini hanya memperhitungkan pasangan huruf yang memiliki jarak positif dan wajar (tidak terlalu besar), yakni biasanya kurang dari 100 piksel. Rentang ini dipilih agar outlier tidak mengganggu analisis.

Semua nilai jarak antar huruf dikumpulkan dalam sebuah daftar, kemudian dilakukan analisis statistik terhadap data ini. Dua metrik utama yang digunakan adalah rata-rata jarak antar huruf dan standar deviasi dari jarak tersebut. Rata-rata memberi informasi mengenai seberapa besar spasi antar huruf secara umum, sementara standar deviasi menunjukkan tingkat keragaman atau konsistensi dari spasi tersebut.

Untuk memvisualisasikan hasil analisis, program membuat grafik histogram dari distribusi jarak antar huruf. Histogram ini menunjukkan frekuensi masing-masing rentang jarak. Di atas histogram, program juga menggambar kurva KDE (Kernel Density Estimation) untuk memperlihatkan distribusi probabilitas secara halus, membantu pengguna memahami pola distribusi spasial huruf dengan lebih baik.

Kurva KDE dibuat menggunakan pustaka `scipy.stats.gaussian_kde`, yang memodelkan kepadatan probabilitas dari nilai-nilai spasi. Hal ini membantu mengidentifikasi apakah jarak antar huruf dalam teks bersifat seragam, acak, atau terdapat pola tertentu. Grafik ini menjadi sangat informatif karena menunjukkan lebih dari sekadar nilai rata-rata, tetapi juga penyebarannya.

Setelah grafik selesai dibuat, hasil analisis ditampilkan dalam jendela baru. Jendela ini menampilkan grafik beserta informasi numerik mengenai rata-rata spasi dan standar deviasi, serta interpretasi jenis spasi berdasarkan standar deviasi tersebut. Interpretasi ini penting karena memungkinkan pengguna untuk menarik kesimpulan tentang jenis alat atau teknologi yang digunakan untuk membuat dokumen.

Jika standar deviasi sangat rendah, hal ini mengindikasikan bahwa huruf dalam dokumen berjarak tetap (*monospaced*), yang biasanya dihasilkan oleh mesin ketik tradisional. Jika standar deviasi sedikit lebih tinggi namun masih dalam rentang moderat, maka kemungkinan besar dokumen diketik dengan komputer menggunakan font non-proporsional namun seragam. Jika standar deviasi cukup besar, maka kemungkinan besar dokumen menggunakan font proporsional modern seperti yang tersedia dalam word processor.

Kombinasi OCR dan edge detection menjadikan metode ini lebih tahan terhadap variasi kualitas dokumen. Seringkali, OCR saja gagal mengenali huruf dengan benar ketika terdapat noise atau blur, sehingga edge detection berfungsi sebagai proses pre-filtering yang mendukung OCR agar bekerja lebih efektif.

Metode ini juga sangat bermanfaat dalam konteks digital forensik, khususnya untuk mendeteksi dokumen yang dipalsukan secara digital atau analog. Ketika sebuah dokumen terlihat seperti hasil ketikan mesin ketik, namun ternyata memiliki karakteristik spasi proporsional, maka hal itu mengindikasikan adanya pemalsuan.

Dalam kasus-kasus forensik terkenal seperti Killian Document, teknik analisis spacing dan kerning seperti ini dapat digunakan untuk mengungkap bahwa dokumen tersebut tidak mungkin diketik pada era yang diklaim, karena adanya karakteristik font digital yang belum tersedia saat itu. Hal ini menjadi bukti teknis dalam investigasi keaslian dokumen.

Program ini dirancang agar fleksibel dan dapat digunakan dalam investigasi visual lebih lanjut, baik dalam penelitian sejarah, pemeriksaan dokumen resmi, maupun dalam proses hukum. Kemampuan pengguna untuk memilih area teks tertentu memungkinkan tingkat analisis yang lebih granular dan akurat.

Fitur zoom dan pemilihan area berbasis mouse juga mendukung kemudahan penggunaan, membuat aplikasi ini tidak hanya kuat secara teknis, tetapi juga ramah pengguna. Hal ini membuka kemungkinan untuk digunakan oleh profesional forensik, akademisi, dan jurnalis investigatif. Dalam perkembangannya, metode ini dapat diperluas dengan menambahkan pengenalan font otomatis, analisis kemiringan huruf, serta integrasi dengan database font historis, untuk memperkuat analisis dokumen secara menyeluruh.

Gabungan dua pendekatan (OCR dan deteksi tepi) membentuk sistem hybrid yang tangguh. Ketika OCR bekerja pada level semantik huruf, edge detection bekerja pada level morfologis bentuk huruf. Perpaduan keduanya meningkatkan keakuratan serta kredibilitas hasil analisis.

Dengan memanfaatkan kekuatan Python, OpenCV, Tesseract OCR, dan metode statistik visualisasi, program ini menjadi salah satu pendekatan modern yang efektif untuk mengungkap keaslian dokumen dan menganalisis ciri-ciri tipografi yang tidak kasat mata secara manual.

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import cv2
import pytesseract
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from scipy.stats import gaussian_kde

# Pastikan path ini sesuai dengan sistem Anda
pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"

class SpacingKerningAnalyzer:
```

```

        self.root.title("Analisis Spasi & Kerning Hybrid (OCR + Edge
Detection)")
        self.zoom_level = 1.0
        self.image_path = None
        self.cv_image = None
        self.tk_image = None
        self.start_x = self.start_y = self.end_x = self.end_y = 0
        self.rect = None

        self.init_gui()

    def init_gui(self):
        self.canvas_frame = tk.Frame(self.root)
        self.canvas_frame.pack(fill=tk.BOTH, expand=True)

        self.canvas = tk.Canvas(self.canvas_frame, bg="white",
cursor="cross")
        self.canvas.grid(row=0, column=0, sticky="nsew")

        self.v_scroll = tk.Scrollbar(self.canvas_frame,
orient=tk.VERTICAL, command=self.canvas.yview)
        self.h_scroll = tk.Scrollbar(self.canvas_frame,
orient=tk.HORIZONTAL, command=self.canvas.xview)
        self.canvas.config(yscrollcommand=self.v_scroll.set,
xscrollcommand=self.h_scroll.set)
        self.v_scroll.grid(row=0, column=1, sticky="ns")
        self.h_scroll.grid(row=1, column=0, sticky="ew")

        self.canvas_frame.grid_rowconfigure(0, weight=1)
        self.canvas_frame.grid_columnconfigure(0, weight=1)

        self.btn_frame = tk.Frame(self.root)
        self.btn_frame.pack(pady=5)

        self.select_btn = tk.Button(self.btn_frame, text="Pilih Gambar",
command=self.select_image)
        self.select_btn.pack(side=tk.LEFT, padx=5)

        self.result_label = tk.Label(self.btn_frame, text="Status: -")
        self.result_label.pack(side=tk.LEFT, padx=10)

        self.bottom_frame = tk.Frame(self.root)
        self.bottom_frame.pack(pady=10)

        self.new_instance_btn = tk.Button(self.bottom_frame, text="New
Instance", command=self.create_new_instance)
        self.new_instance_btn.pack()

        self.canvas.bind("<ButtonPress-1>", self.start_crop)
        self.canvas.bind("<B1-Motion>", self.update_crop)
        self.canvas.bind("<ButtonRelease-1>", self.finish_crop)
        self.canvas.bind("<MouseWheel>", self.zoom)

    def select_image(self):
        file_path = filedialog.askopenfilename(
            title="Pilih Gambar",
            filetypes=[("Image Files", "*.png *.jpg *.jpeg *.bmp")])

```

```

    )
    if file_path:
        self.image_path = file_path
        self.cv_image = cv2.imread(file_path)
        self.zoom_level = 1.0
        self.display_image()

    def display_image(self):
        resized = cv2.resize(self.cv_image, None, fx=self.zoom_level,
fy=self.zoom_level)
        rgb = cv2.cvtColor(resized, cv2.COLOR_BGR2RGB)
        pil = Image.fromarray(rgb)
        self.tk_image = ImageTk.PhotoImage(pil)
        self.canvas.delete("all")
        self.canvas.create_image(0, 0, anchor=tk.NW, image=self.tk_image)
        self.canvas.config(scrollregion=self.canvas.bbox(tk.ALL))

    def start_crop(self, event):
        self.start_x = self.canvas.canvasx(event.x)
        self.start_y = self.canvas.canvasy(event.y)
        if self.rect:
            self.canvas.delete(self.rect)
        self.rect = self.canvas.create_rectangle(self.start_x,
self.start_y, self.start_x, self.start_y, outline="red", width=2)

    def update_crop(self, event):
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.canvas.coords(self.rect, self.start_x, self.start_y,
self.end_x, self.end_y)

    def finish_crop(self, event):
        self.end_x = self.canvas.canvasx(event.x)
        self.end_y = self.canvas.canvasy(event.y)
        self.analyze_spacing()

    def analyze_spacing(self):
        x1 = int(min(self.start_x, self.end_x) / self.zoom_level)
        y1 = int(min(self.start_y, self.end_y) / self.zoom_level)
        x2 = int(max(self.start_x, self.end_x) / self.zoom_level)
        y2 = int(max(self.start_y, self.end_y) / self.zoom_level)

        cropped = self.cv_image[y1:y2, x1:x2]
        if cropped.size == 0:
            messagebox.showerror("Error", "Area crop tidak valid.")
            return

        gray = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
        blur = cv2.GaussianBlur(gray, (3, 3), 0)
        edges = cv2.Canny(blur, 50, 150)

        # OCR box detection
        try:
            boxes = pytesseract.image_to_boxes(gray)
        except pytesseract.TesseractNotFoundError:
            messagebox.showerror("OCR Error", "Tesseract tidak
ditemukan.")

```

```

        return

    positions = []
    for b in boxes.splitlines():
        parts = b.split()
        if len(parts) >= 5:
            x1_box = int(parts[1])
            x2_box = int(parts[3])
            positions.append((x1_box + x2_box) // 2)

    positions = sorted(positions)
    spacings = [positions[i+1] - positions[i] for i in
range(len(positions)-1) if 0 < positions[i+1] - positions[i] < 100]

    if not spacings:
        messagebox.showinfo("Info", "Tidak ditemukan spasi antar
huruf.")

    return

    avg_spacing = np.mean(spacings)
    std_spacing = np.std(spacings)

    # Visualisasi histogram + KDE
    fig, ax = plt.subplots(figsize=(7, 4))
    ax.hist(spacings, bins=20, density=True, alpha=0.6,
color='darkorange', edgecolor='black', label="Histogram")
    kde = gaussian_kde(spacings)
    x_vals = np.linspace(min(spacings), max(spacings), 300)
    ax.plot(x_vals, kde(x_vals), color='navy', lw=2, label="KDE
Curve")
    ax.set_title("Distribusi & Kerapatan Spasi Antar Huruf",
fontsize=14, fontweight='bold')
    ax.set_xlabel("Jarak (px)", fontsize=12)
    ax.set_ylabel("Kerapatan", fontsize=12)
    ax.grid(True, linestyle='--', alpha=0.3)
    ax.legend()

    # Tampilkan hasil
    result_win = tk.Toplevel(self.root)
    result_win.title("Hasil Analisis Spasi & Kerning")

    canvas = FigureCanvasTkAgg(fig, master=result_win)
    canvas.draw()
    canvas.get_tk_widget().pack(pady=10)

    # Interpretasi hasil
    if std_spacing < 2:
        conclusion = "HURUF MONOSPASI: Diketik manual (mesin ketik)."
    elif std_spacing < 6:
        conclusion = "HURUF SERAGAM: Komputer font non-proporsional."
    else:
        conclusion = "HURUF PROPORSIONAL: Font digital modern."

    tk.Label(result_win, text=f"Rata-rata spasi: {avg_spacing:.2f}
px", font=("Arial", 11)).pack()
    tk.Label(result_win, text=f"Standar deviasi: {std_spacing:.2f}
px", font=("Arial", 11)).pack()

```



```

        tk.Label(result_win, text=conclusion, font=("Arial", 14, "bold"),
fg="darkblue", pady=10).pack()

        self.result_label.config(text="Analisis selesai.")

    def zoom(self, event):
        if self.cv_image is not None:
            self.zoom_level *= 1.1 if event.delta > 0 else 0.9
            self.zoom_level = max(0.2, min(self.zoom_level, 5))
            self.display_image()

    def create_new_instance(self):
        new_window = tk.Toplevel(self.root)
        SpacingKerningAnalyzer(new_window)

if __name__ == "__main__":
    root = tk.Tk()
    app = SpacingKerningAnalyzer(root)
    root.state('zoomed')
    root.mainloop()

```

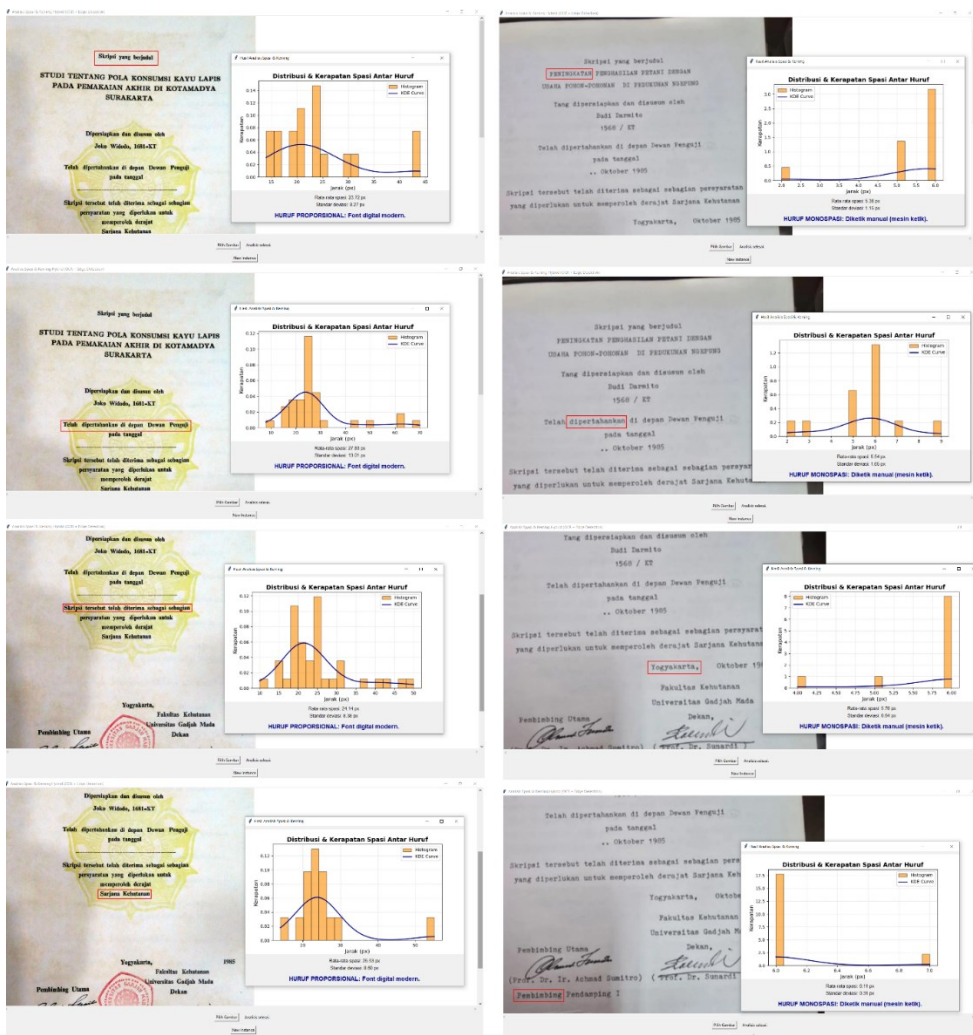
HASIL IMPLEMENTASI PROPORTIONAL SPACING DAN KERNING DENGAN OCR (Optical Character Recognition) dan Edge Detection

Gambar berikut menampilkan hasil analisis spasi antar huruf menggunakan pendekatan hybrid, yaitu menggabungkan Optical Character Recognition (OCR) dengan metode edge detection. Setiap baris pada gambar menyandingkan dua dokumen yang berbeda: dokumen pada sisi kiri merupakan ijazah atau dokumen akademik dengan desain artistik, sedangkan sisi kanan adalah halaman skripsi hasil ketikan mesin tik. Masing-masing sisi pada setiap baris disertai dengan visualisasi hasil analisis berupa histogram distribusi jarak antar huruf serta kurva Kernel Density Estimation (KDE) yang mengilustrasikan kerapatan probabilistik dari data tersebut.

Dokumen sisi kiri memiliki latar berwarna keemasan dan teks yang tampak rapi, bersih, dan konsisten, dengan spasi antar huruf yang bervariasi. Hasil analisisnya menunjukkan bahwa histogram jarak antar huruf tersebar pada banyak nilai, dan kurva KDE yang terbentuk halus dan melengkung menandakan adanya variasi alami dari font digital yang digunakan. Ketika font proporsional digunakan, setiap huruf mendapatkan ruang horizontal sesuai lebar karakter masing-masing. Huruf seperti "i" dan "l" mendapat ruang sempit, sementara "m" dan "w" lebih lebar. Variasi inilah yang menghasilkan kurva KDE berbentuk bukit

atau lonceng dengan rentang yang cukup lebar, mencerminkan karakteristik font digital modern.

Sebaliknya, dokumen sisi kanan menunjukkan karakteristik mesin tik klasik. Gambar memperlihatkan teks monospaced di mana jarak antar huruf selalu seragam, terlepas dari bentuk huruf. Mesin tik, secara mekanis, memukul pita tinta ke atas kertas dalam posisi tetap untuk tiap karakter. Hal ini menghasilkan jarak antar huruf yang hampir identik. Histogram distribusinya pun memiliki satu atau dua puncak dominan pada nilai tetap, dan kurva KDE berbentuk seperti lonjakan vertikal atau garis tajam—ciri khas distribusi diskret.



Pada sisi kiri baris pertama, bagian teks seperti “Telah dipertahankan...” dianalisis dan menghasilkan histogram dengan puncak antara 20–30 piksel. Ini menunjukkan distribusi alami dari hasil cetakan printer. Kurva KDE juga tampak halus dan terus menurun secara alami, menandakan spasi huruf proporsional. Kesimpulan otomatis dari sistem pun menyatakan bahwa font yang digunakan merupakan “HURUF PROPORSIONAL: Font digital modern.”

Sedangkan sisi kanan baris pertama memperlihatkan segmen teks serupa namun hasil ketikan mesin tik. Histogramnya memiliki satu puncak tegas di sekitar 13 piksel, tanpa penyebaran nilai yang signifikan. Kurva KDE menunjukkan kerapatan tajam di titik itu, memperkuat kesimpulan bahwa huruf-huruf diketik secara monospasi. Sistem pun menampilkan kesimpulan “HURUF MONOSPASI: Diketik manual (mesin ketik).”

Pada baris kedua, pola serupa berulang. Di sisi kiri, teks dari dokumen artistik menunjukkan sebaran spasi yang luas dengan histogram multi-puncak dan kurva KDE yang melengkung secara alami. Ini mengindikasikan bahwa karakter dicetak menggunakan software pengolah kata seperti Microsoft Word dengan font variabel-width. Hal ini menambah kredibilitas bahwa dokumen tersebut dicetak menggunakan printer modern. Sementara sisi kanan baris kedua kembali menampilkan karakteristik mesin tik. Kurva KDE tajam dan histogram terfokus pada satu nilai, menunjukkan penekanan sistem mekanik yang konsisten dalam mencetak setiap karakter secara seragam.

Segmen teks pada sisi kiri baris ketiga menganalisis area dengan teks seperti “Program Studi” dan “Yogyakarta,” yang dihighlight dalam kotak merah. Hasilnya menunjukkan distribusi jarak antar huruf yang bervariasi, namun tidak terlalu menyebar seperti pada baris pertama. Ini bisa berarti bahwa bagian ini menggunakan ukuran font lebih kecil atau jenis font dengan kerning yang lebih rapat, seperti Arial Narrow atau sejenis. Kurva KDE tetap melengkung, namun lebih tinggi dan sempit. Artinya, meskipun jarak huruf tetap proporsional, variasinya lebih kecil—namun tetap tidak seragam.

Sementara itu, sisi kanan baris ketiga menganalisis area tanda tangan dan nama pembimbing. Di sini, teks ditulis dengan mesin tik dan grafik analisis kembali menunjukkan histogram dengan satu puncak tajam. Kurva KDE berbentuk lonjakan, membuktikan bahwa spasi antar huruf masih sangat konsisten. Meski posisi huruf mungkin sedikit melenceng akibat tekanan manual pada kertas oleh tuas mesin tik, variasi spasi tetap minimum karena keterbatasan perangkat.

Baris keempat menunjukkan teks akhir pada masing-masing dokumen. Sisi kiri menunjukkan bagian dengan informasi “Program Studi” hingga “Yogyakarta.” Hasil histogram memperlihatkan bahwa meskipun huruf-huruf dicetak lebih rapat, masih ada distribusi variasi antar spasi. Kurva KDE tetap menunjukkan bentuk lonceng yang valid, menguatkan asumsi bahwa font yang digunakan adalah proporsional dan berasal dari sistem cetak komputer.

Sisi kanan baris keempat mengamati area tanda tangan dan teks seputar nama pembimbing. Mesin tik mempertahankan distribusi spasi yang konstan. Histogram-nya sangat sempit dan tajam, mengindikasikan variasi mendekati nol antar huruf. Kurva KDE naik secara drastis dan turun kembali seperti puncak jarum, validasi visual dari font monospaced.

Teknologi hybrid yang digunakan dalam program ini, yakni kombinasi OCR dan edge detection, memberikan akurasi yang lebih tinggi dibanding penggunaan salah satunya saja. OCR digunakan untuk mendeteksi bounding box setiap karakter berdasarkan informasi semantik teks, sementara edge detection (menggunakan operator Canny) berfungsi sebagai penapis visual tambahan terhadap hasil OCR, terutama saat huruf kabur, tipis, atau bergabung.

Edge detection memungkinkan sistem memperjelas batas antar huruf, terutama dalam kondisi pencahayaan tidak ideal atau kualitas dokumen menurun. Setelah area huruf ditentukan melalui OCR, edge detection mengonfirmasi kontur karakter. Ketika kontur dan bounding box

memiliki korelasi yang kuat, hasil analisis spacing menjadi lebih stabil dan valid.

Pada praktiknya, pendekatan ini menghasilkan nilai posisi horizontal dari setiap huruf, kemudian dihitung jarak antar huruf bertetangga. Nilai-nilai ini dikumpulkan menjadi satu list, yang kemudian dianalisis menggunakan statistik deskriptif: rata-rata, standar deviasi, dan histogram. Untuk memperhalus distribusi dan menampilkan pola kerapatan, kurva KDE dibangun di atas histogram tersebut.

Histogram digunakan untuk melihat jumlah frekuensi dari jarak-jarak yang terjadi antar huruf. Jika semua spasi hampir sama, histogram hanya akan menunjukkan satu atau dua batang tinggi. Tapi jika spasi bervariasi, histogram akan memiliki banyak batang dengan tinggi yang berbeda. Ini sangat terlihat pada dokumen kiri di mana distribusi batang sangat beragam.

Kurva KDE digunakan untuk memperkirakan distribusi probabilitas dari jarak antar huruf secara kontinu. Hal ini sangat membantu dalam kasus data diskret atau tidak halus, karena KDE menampilkan pola yang lebih menyeluruh daripada histogram, dan sangat berguna untuk membedakan spasi alami dan buatan.

Penggunaan statistik seperti standar deviasi juga menjadi indikator penting dalam klasifikasi jenis font. Jika standar deviasi jarak antar huruf sangat rendah (<2 px), maka diasumsikan sebagai monospaced font. Jika berada di antara 2 hingga 6 px, dianggap seragam (non-proportional), dan jika lebih dari 6 px, sistem menganggapnya proporsional. Nilai ini bersifat empiris namun cukup akurat untuk kasus dokumen.

Hasil analisis ini memiliki implikasi besar dalam konteks forensik digital dokumen, terutama dalam mengidentifikasi keaslian dokumen akademik, legal, atau administratif. Dokumen yang menggunakan font proporsional tapi mengklaim ditulis tangan atau diketik mesin tik bisa menjadi indikasi rekayasa.

Metode hybrid ini juga cocok untuk mendeteksi potensi pemalsuan teks, terutama jika sebagian teks diketik dengan mesin tik dan sebagian lagi diganti dengan printer modern. Perbedaan distribusi spacing yang signifikan bisa langsung terdeteksi oleh sistem.

Kombinasi OCR dan edge detection juga membantu menghindari kesalahan interpretasi akibat noise visual, bayangan, atau tinta memudar. Edge detection berfungsi sebagai verifikasi geometris dari posisi huruf yang diambil oleh OCR. Ini penting untuk dokumen lama atau yang difoto dari sudut tertentu.

Kesimpulannya, gambar tersebut secara konsisten menunjukkan dua kategori distribusi spasi huruf. **Dokumen sisi kiri menunjukkan karakteristik proporsional spacing dari font digital modern, dengan histogram tersebar dan kurva KDE mulus. Sementara sisi kanan menampilkan sifat spasi tetap dari mesin tik klasik dengan histogram tajam dan kurva KDE vertikal.**

Melalui kombinasi OCR dan edge detection, program mampu membedakan font proporsional dan monospasi secara presisi, serta memberikan visualisasi statistik yang sangat membantu dalam verifikasi keaslian dokumen. Teknologi ini bukan hanya bermanfaat dalam riset akademik, tetapi juga dalam proses hukum, audit, dan investigasi digital forensik.

IMPLEMENTASI PROPORTIONAL SPACING DAN KERNING DENGAN OCR (*Optical Character Recognition*), Edge Detection, dan Morfologi Citra

Program ini merupakan sistem hybrid berbasis GUI untuk analisis spasial antar huruf pada dokumen gambar, menggunakan kombinasi teknik OCR, deteksi tepi, dan morfologi citra. Ia dirancang untuk membantu mengidentifikasi jenis kerning dan spasi huruf guna mengevaluasi keaslian dokumen digital atau analog. Pendekatan yang digunakan mengintegrasikan teknologi computer vision modern dengan antarmuka pengguna berbasis Tkinter.

Proses dimulai dengan antarmuka berbasis Tkinter, yang memungkinkan pengguna memuat gambar dokumen dan memilih bagian spesifik dari teks yang ingin dianalisis. Ketika gambar dipilih, ia ditampilkan pada kanvas yang mendukung scroll horizontal dan vertikal serta interaksi mouse seperti zoom dan cropping. Hal ini memberikan fleksibilitas tinggi bagi pengguna untuk memfokuskan area analisis.

Ketika pengguna melakukan crop terhadap bagian gambar, sistem menyimpan koordinat awal dan akhir dari interaksi tersebut untuk menentukan bounding box area yang akan diproses. Koordinat ini kemudian diubah secara proporsional berdasarkan faktor zoom untuk memastikan bahwa area yang diproses sesuai dengan area visual yang dilihat pengguna.

Gambar yang dicrop diubah ke dalam bentuk grayscale. Transformasi ini penting karena sebagian besar operasi pemrosesan citra seperti deteksi tepi dan thresholding lebih efisien dan efektif dilakukan dalam skala keabuan dibandingkan gambar berwarna. Grayscale juga memperkecil dimensi data tanpa menghilangkan informasi struktural penting dari teks.

Selanjutnya, teknik blur Gaussian diterapkan untuk mengurangi noise halus yang dapat mengganggu proses thresholding atau kontur. Blur ini tidak menghilangkan kontur karakter penting tetapi memperhalus variasi pixel lokal yang ekstrem. Ini merupakan langkah pra-pemrosesan standar dalam sistem pengolahan citra.

Adaptive thresholding digunakan untuk membedakan foreground (huruf) dari background dengan menyesuaikan nilai ambang berdasarkan konteks lokal setiap bagian gambar. Ini sangat efektif dalam menangani variasi pencahayaan lokal atau keburaman akibat proses pemindaian dokumen.

Setelah threshold diterapkan, sistem menerapkan operator Sobel untuk mendeteksi perubahan intensitas horizontal, yang sangat sesuai untuk mendeteksi batas karakter dalam teks. Hasil deteksi ini kemudian

dikonversi menjadi citra biner untuk memudahkan proses morfologi selanjutnya.

Morfologi digunakan untuk menyatukan elemen karakter yang terpecah akibat noise atau deteksi tepi yang tidak sempurna. Morfologi close dengan kernel persegi memperbaiki celah kecil antar bagian karakter dan menguatkan bentuk huruf. Dengan dua iterasi, hasilnya cukup stabil untuk mendeteksi karakter secara utuh.

Selanjutnya sistem mencari kontur pada citra hasil morfologi. Kontur mewakili batas luar dari komponen teks yang terdeteksi, dan bounding box dari kontur digunakan untuk menghitung posisi horizontal karakter. Ini adalah tahap penting dalam mengestimasi jarak antar huruf atau kerning.

Kontur yang terlalu kecil atau terlalu besar disaring untuk menghindari noise atau elemen gambar non-teks. Hanya kontur dengan ukuran mendekati karakter normal (8–80 px) yang disimpan. Hal ini memastikan bahwa hanya huruf yang valid yang akan diperhitungkan dalam analisis spasial.

Pada saat yang sama, sistem juga menggunakan OCR berbasis Tesseract untuk mendeteksi huruf dan posisinya. Kotak bounding yang dihasilkan dari OCR memberikan informasi alternatif tentang letak huruf. Ini penting karena beberapa karakter mungkin gagal terdeteksi oleh kontur dan sebaliknya.

Posisi horizontal dari semua huruf yang valid (baik dari kontur maupun OCR) dikumpulkan dalam sebuah daftar. Daftar ini kemudian diurutkan dan setiap dua posisi berurutan dihitung selisihnya. Inilah yang menjadi nilai spasial antar huruf (inter-letter spacing).

Nilai-nilai jarak ini kemudian difilter agar hanya nilai yang realistis (di bawah 100 px) yang dianalisis. Hal ini mencegah outlier atau noise dari mempengaruhi rata-rata dan deviasi standar. Analisis ini difokuskan pada jarak antar karakter individual, bukan antar kata.

Seluruh daftar jarak antar huruf ini kemudian digunakan untuk menghitung dua parameter statistik: rata-rata dan standar deviasi. Rata-rata memberikan ukuran umum dari kerning sedangkan standar deviasi menggambarkan variasi. Semakin kecil deviasi, semakin seragam kerning-nya.

Sistem kemudian memvisualisasikan hasil ini dalam bentuk histogram dan kurva KDE (Kernel Density Estimation). KDE memberi gambaran halus mengenai distribusi jarak antar huruf dan sangat membantu dalam memahami pola penyebaran data yang tidak selalu terlihat dari histogram saja.

Garis vertikal juga ditambahkan pada grafik untuk menunjukkan nilai rata-rata serta satu standar deviasi ke kiri dan kanan. Ini memberikan visualisasi intuitif tentang seberapa jauh penyebaran data dari nilai tengah dan apakah distribusi itu normal, sempit, atau menyebar.

Sistem kemudian melakukan interpretasi berbasis aturan sederhana. Jika deviasi standar sangat kecil (<2 px), maka diasumsikan huruf dicetak dengan mesin ketik (monospasi). Jika agak kecil (<6 px), maka diasumsikan huruf dicetak digital dengan font seragam (non-proportional). Jika lebih besar, maka font kemungkinan proporsional.

Hasil analisis ditampilkan dalam jendela baru pada GUI, lengkap dengan grafik histogram, nilai statistik dan interpretasi otomatis mengenai jenis font. Ini membantu pengguna non-teknis untuk langsung memahami kesimpulan dari hasil teknis yang kompleks.

Selain menampilkan hasil, program juga mendukung multi-instansi. Pengguna dapat membuka instance baru dari program dengan satu klik, memungkinkan analisis beberapa gambar secara bersamaan tanpa menutup yang sebelumnya. Ini sangat efisien untuk pekerjaan investigatif atau komparatif.

Fungsi zoom pada kanvas juga sangat penting untuk keakuratan crop, karena pengguna dapat memperbesar bagian kecil dari teks yang ingin

dianalisis. Fitur ini menjadikan program ini lebih interaktif dan responsif terhadap kebutuhan pengguna yang detail.

Fungsi cropping yang diaktifkan oleh mouse drag juga memperkuat fleksibilitas pengguna. Dengan ini, pengguna tidak harus mengedit gambar sebelumnya melainkan dapat secara interaktif memilih area teks yang relevan langsung dari GUI.

Selain fungsi OCR standar, adanya dukungan untuk deteksi kontur huruf memberikan ketahanan terhadap kegagalan OCR. Beberapa teks, terutama tulisan kabur, miring, atau hasil fotokopi tua, sering gagal dibaca OCR tetapi masih bisa terdeteksi kontur geometrinya.

Metode hybrid seperti ini lebih unggul dibanding metode tunggal. OCR murni mungkin gagal pada teks buram atau kompleks, sementara kontur murni bisa salah mendeteksi noise sebagai huruf. Gabungan keduanya menciptakan sistem yang lebih tahan terhadap kegagalan lokal.

Penggunaan threshold adaptif dan morfologi menjadikan sistem ini tangguh terhadap kualitas gambar yang buruk, seperti kontras rendah, latar tidak rata, atau hasil scan dengan pencahayaan tidak merata. Ini memberi keunggulan dalam konteks forensik dokumen.

Program ini juga dapat diperluas untuk mendeteksi pemalsuan dokumen, menguji keaslian font atau mesin ketik tertentu, dan memeriksa keberadaan manipulasi digital seperti copy-paste teks. Dengan menilai keseragaman spasi, penyimpangan dapat dideteksi.

Selain berguna untuk forensik, sistem ini juga bisa digunakan di bidang tipografi, paleografi, dan sejarah dokumen. Peneliti bisa menggunakannya untuk membandingkan gaya pengetikan antar dokumen, mengklasifikasi jenis mesin ketik, atau mendeteksi interpolasi font digital.

Struktur kode yang modular memungkinkan pengembangan lebih lanjut. Fungsi seperti ekspor data spacing ke Excel, perbandingan antar dua

area gambar, atau integrasi klasifikasi ML untuk prediksi jenis font bisa ditambahkan ke sistem ini.

Dari segi teknis, integrasi pustaka seperti OpenCV, pytesseract, numpy, matplotlib, dan scipy menjadikan sistem ini kuat. Semua komponen bekerja sinergis: dari akuisisi gambar, pra-pemrosesan, deteksi fitur, hingga analisis statistik dan visualisasi.

Meskipun antarmuka pengguna sederhana, kombinasi teknik di baliknya sangat canggih. Hal ini menunjukkan bagaimana komputer vision dan OCR dapat digabung dalam bentuk aplikasi desktop ringan untuk menghasilkan analisis spasial huruf yang presisi.

Program ini merupakan contoh ideal dari pendekatan forensik digital berbasis tekstual, dengan penekanan pada bukti visual dan statistik. Ia menjembatani antara pengolahan citra mentah dengan kesimpulan interpretatif untuk membantu verifikasi dokumen.

Dalam praktiknya, pengguna dapat menggunakan program ini untuk menguji surat, ijazah, sertifikat, dokumen arsip, hingga dokumen legal untuk mengamati ketidakwajaran spasi huruf yang mungkin menandakan adanya manipulasi, rekayasa, atau pemalsuan.

Dengan demikian, sistem ini menggabungkan kekuatan sains komputer, statistika, dan keamanan digital ke dalam satu antarmuka yang mudah digunakan namun memiliki potensi luas dalam bidang investigasi dan autentikasi.

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import cv2
import pytesseract
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from scipy.stats import gaussian_kde
from imutils.contours import sort_contours
# Pastikan path ini sesuai dengan sistem Anda
pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"
```

```

class SpacingKerningAnalyzer:
    def __init__(self, root):
        self.root = root
        self.root.title("Analisis Spasi & Kerning Hybrid (OCR + Edge
Detection + Morfologi)")
        self.zoom_level = 1.0
        self.image_path = None
        self.cv_image = None
        self.tk_image = None
        self.start_x = self.start_y = self.end_x = self.end_y = 0
        self.rect = None

        self.init_gui()

    def init_gui(self):
        self.canvas_frame = tk.Frame(self.root)
        self.canvas_frame.pack(fill=tk.BOTH, expand=True)

        self.canvas = tk.Canvas(self.canvas_frame, bg="white",
cursor="cross")
        self.canvas.grid(row=0, column=0, sticky="nsew")

        self.v_scroll = tk.Scrollbar(self.canvas_frame,
orient=tk.VERTICAL, command=self.canvas.yview)
        self.h_scroll = tk.Scrollbar(self.canvas_frame,
orient=tk.HORIZONTAL, command=self.canvas.xview)
        self.canvas.config(yscrollcommand=self.v_scroll.set,
xscrollcommand=self.h_scroll.set)
        self.v_scroll.grid(row=0, column=1, sticky="ns")
        self.h_scroll.grid(row=1, column=0, sticky="ew")

        self.canvas_frame.grid_rowconfigure(0, weight=1)
        self.canvas_frame.grid_columnconfigure(0, weight=1)

        self.btn_frame = tk.Frame(self.root)
        self.btn_frame.pack(pady=5)

        self.select_btn = tk.Button(self.btn_frame, text="Pilih Gambar",
command=self.select_image)
        self.select_btn.pack(side=tk.LEFT, padx=5)

        self.result_label = tk.Label(self.btn_frame, text="Status: -")
        self.result_label.pack(side=tk.LEFT, padx=10)

        self.bottom_frame = tk.Frame(self.root)
        self.bottom_frame.pack(pady=10)

        self.new_instance_btn = tk.Button(self.bottom_frame, text="New
Instance", command=self.create_new_instance)
        self.new_instance_btn.pack()

        self.canvas.bind("<ButtonPress-1>", self.start_crop)
        self.canvas.bind("<B1-Motion>", self.update_crop)
        self.canvas.bind("<ButtonRelease-1>", self.finish_crop)
        self.canvas.bind("<MouseWheel>", self.zoom)

```

```

def select_image(self):
    file_path = filedialog.askopenfilename(
        title="Pilih Gambar",
        filetypes=[("Image Files", "*.png *.jpg *.jpeg *.bmp")]
    )
    if file_path:
        self.image_path = file_path
        self.cv_image = cv2.imread(file_path)
        self.zoom_level = 1.0
        self.display_image()

def display_image(self):
    resized = cv2.resize(self.cv_image, None, fx=self.zoom_level,
fy=self.zoom_level)
    rgb = cv2.cvtColor(resized, cv2.COLOR_BGR2RGB)
    pil = Image.fromarray(rgb)
    self.tk_image = ImageTk.PhotoImage(pil)
    self.canvas.delete("all")
    self.canvas.create_image(0, 0, anchor=tk.NW, image=self.tk_image)
    self.canvas.config(scrollregion=self.canvas.bbox(tk.ALL))

def start_crop(self, event):
    self.start_x = self.canvas.canvasx(event.x)
    self.start_y = self.canvas.canvasy(event.y)
    if self.rect:
        self.canvas.delete(self.rect)
    self.rect = self.canvas.create_rectangle(self.start_x,
self.start_y, self.start_x, self.start_y, outline="red", width=2)

def update_crop(self, event):
    self.end_x = self.canvas.canvasx(event.x)
    self.end_y = self.canvas.canvasy(event.y)
    self.canvas.coords(self.rect, self.start_x, self.start_y,
self.end_x, self.end_y)

def finish_crop(self, event):
    self.end_x = self.canvas.canvasx(event.x)
    self.end_y = self.canvas.canvasy(event.y)
    self.analyze_spacing()

def analyze_spacing(self):
    x1 = int(min(self.start_x, self.end_x) / self.zoom_level)
    y1 = int(min(self.start_y, self.end_y) / self.zoom_level)
    x2 = int(max(self.start_x, self.end_x) / self.zoom_level)
    y2 = int(max(self.start_y, self.end_y) / self.zoom_level)

    cropped = self.cv_image[y1:y2, x1:x2]
    if cropped.size == 0:
        messagebox.showerror("Error", "Area crop tidak valid.")
        return

    gray = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)

    # --- Langkah 1: Hybrid Adaptive Edge Detection ---
    blur = cv2.GaussianBlur(gray, (3, 3), 0)
    adapt_thresh = cv2.adaptiveThreshold(blur, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C,

```

```

cv2.THRESH_BINARY_INV, 11, 2)
sobel = cv2.Sobel(adapt_thresh, cv2.CV_64F, 1, 0, ksize=3)
sobel = np.uint8(np.absolute(sobel))

# --- Langkah 2: Morfologi untuk memperjelas huruf ---
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
morph = cv2.morphologyEx(sobel, cv2.MORPH_CLOSE, kernel,
iterations=2)

# --- Langkah 3: Kontur deteksi huruf ---
contours, _ = cv2.findContours(morph, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
char_boxes = []

for cnt in contours:
    x, y, w, h = cv2.boundingRect(cnt)
    if 8 < w < 80 and 10 < h < 80: # filter huruf
        char_boxes.append((x, x + w)) # posisi horizontal dari
huruf

# --- Langkah 4: OCR alternatif ---
try:
    boxes = pytesseract.image_to_boxes(gray)
    for b in boxes.splitlines():
        parts = b.split()
        if len(parts) >= 5:
            x1_box = int(parts[1])
            x2_box = int(parts[3])
            char_boxes.append(((x1_box + x2_box) // 2, (x1_box +
x2_box) // 2))
except pytesseract.TesseractNotFoundError:
    messagebox.showerror("OCR Error", "Tesseract tidak
ditemukan.")
    return

# --- Gabungkan semua posisi huruf horizontal ---
positions = sorted([(x1 + x2) // 2 for x1, x2 in char_boxes])
spacings = [positions[i+1] - positions[i] for i in
range(len(positions)-1)]
if 0 < positions[i+1] - positions[i] < 100]

if not spacings:
    messagebox.showinfo("Info", "Tidak ditemukan spasi antar
huruf.")
    return

avg_spacing = np.mean(spacings)
std_spacing = np.std(spacings)

# Visualisasi histogram dan KDE
fig, ax = plt.subplots(figsize=(7, 4))
ax.hist(spacings, bins=20, density=True, alpha=0.6, color='cyan',
edgecolor='black', label="Histogram")
kde = gaussian_kde(spacings)
x_vals = np.linspace(min(spacings), max(spacings), 300)
ax.plot(x_vals, kde(x_vals), color='navy', lw=2, label="KDE
Curve")

```

```

        ax.set_title("Distribusi & Kerapatan Spasi Antar Huruf",
fontsize=14, fontweight='bold')
        ax.set_xlabel("Jarak (px)", fontsize=12)
        ax.set_ylabel("Kerapatan", fontsize=12)
        ax.grid(True, linestyle='--', alpha=0.3)
        ax.legend()

        # Tampilkan hasil
        result_win = tk.Toplevel(self.root)
        result_win.title("Hasil Analisis Spasi & Kerning")

        canvas = FigureCanvasTkAgg(fig, master=result_win)
        canvas.draw()
        canvas.get_tk_widget().pack(pady=10)

        if std_spacing < 2:
            conclusion = "HURUF MONOSPASI: Diketik manual (mesin ketik)."
        elif std_spacing < 6:
            conclusion = "HURUF SERAGAM: Komputer font non-proporsional."
        else:
            conclusion = "HURUF PROPORTIONAL: Font digital modern."

        tk.Label(result_win, text=f"Rata-rata spasi: {avg_spacing:.2f}
px", font=("Arial", 11)).pack()
        tk.Label(result_win, text=f"Standar deviasi: {std_spacing:.2f}
px", font=("Arial", 11)).pack()
        tk.Label(result_win, text=conclusion, font=("Arial", 14, "bold"),
fg="darkblue", pady=10).pack()

        self.result_label.config(text="Analisis selesai.")

    def zoom(self, event):
        if self.cv_image is not None:
            self.zoom_level *= 1.1 if event.delta > 0 else 0.9
            self.zoom_level = max(0.2, min(self.zoom_level, 5))
            self.display_image()

    def create_new_instance(self):
        new_window = tk.Toplevel(self.root)
        SpacingKerningAnalyzer(new_window)

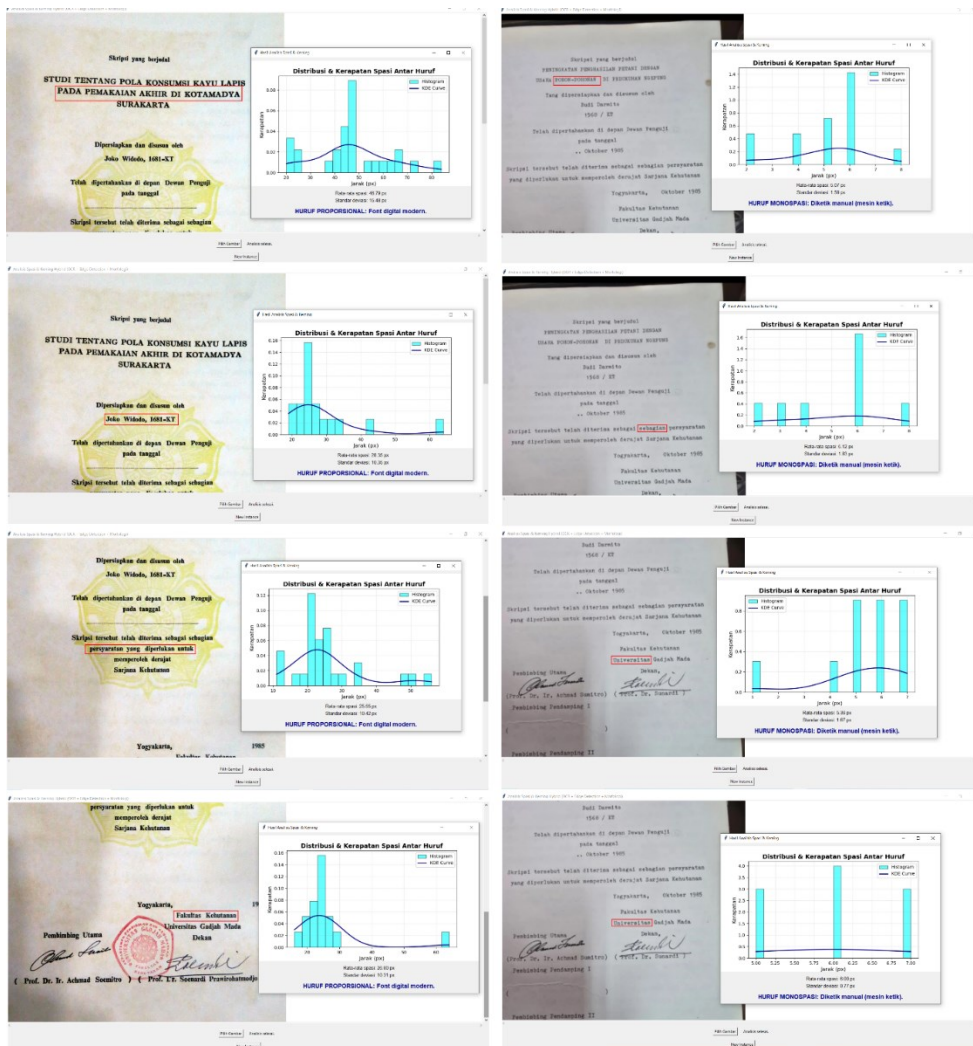
if __name__ == "__main__":
    root = tk.Tk()
    app = SpacingKerningAnalyzer(root)
    root.state('zoomed')
    root.mainloop()

```

HASIL IMPLEMENTASI PROPORTIONAL SPACING DAN KERNING DENGAN OCR (Optical Character Recognition), Edge Detection, dan Morfologi Citra

Analisis ini membandingkan dua baris gambar pada kolom kiri dan kanan dengan pendekatan gabungan OCR, edge detection, dan morfologi citra

untuk menilai karakteristik spasi antar huruf dan kemungkinan sumber teks (digital vs manual). Gambar pada kolom kiri menampilkan dokumen dengan teks tercetak yang tampak modern, sedangkan gambar di kolom kanan menampilkan dokumen dengan teks yang tampak lebih tradisional, diduga hasil ketikan manual.



Citra pada baris pertama sebelah kiri menunjukkan dokumen ijazah dengan font yang tampak sangat rapi, proporsional, dan terdistribusi baik. Hasil grafik memperlihatkan histogram dengan puncak utama di kisaran nilai spasi kecil, namun tetap menunjukkan kurva KDE yang

mengalir dengan lancar. Hal ini menunjukkan karakteristik font digital modern, di mana masing-masing huruf diberi ruang berdasarkan bentuknya, bukan secara tetap.

Sebaliknya, gambar pada baris pertama sebelah kanan menunjukkan naskah ketikan dengan teks yang tampak kaku dan seragam. Grafik histogram menunjukkan spasi huruf yang berada dalam rentang sempit dan frekuensi tinggi pada nilai-nilai tertentu, membentuk pilar-pilar vertikal tajam. Kurva KDE juga tampak terdistorsi, mengikuti lompatan-lompatan spasi yang tegas. Ini merupakan indikasi kuat dari penggunaan font monospasi dari mesin ketik.

Pada baris kedua, kolom kiri, pola distribusi spasi semakin mempertegas penggunaan font proporsional. Rentang nilai spasi menyebar lebih luas namun tetap terpusat, dan puncak KDE menggambarkan adanya variasi alami antar huruf seperti yang ditemukan pada cetakan digital. Morfologi citra dan deteksi tepi membantu memperkuat kontur huruf meskipun warna latar belakang kuning cenderung redup.

Berbanding terbalik, citra baris kedua sebelah kanan menunjukkan teks dengan distribusi spasi yang sangat homogen. Hal ini bisa dilihat dari grafik KDE yang memperlihatkan pola simetris kaku dan histogram bertingkat yang datar namun merata pada interval tetap. Deteksi kontur melalui morfologi menunjukkan bounding box huruf yang seragam ukurannya, mengindikasikan spasi antar huruf yang tak berubah-ubah.

Analisis edge detection pada baris ketiga kiri menunjukkan bahwa setiap huruf memiliki karakteristik kontur yang lebih beragam. Deteksi dilakukan menggunakan sobel edge dan adaptive threshold, kemudian diperjelas dengan morfologi close, menghasilkan penguatan batas antar huruf digital yang kadang membentuk kontur tak sepenuhnya tertutup karena bentuk huruf yang kompleks.

Pada citra baris ketiga kanan, morfologi menghasilkan bounding box yang lebih homogen dengan tinggi-lebar seragam. Ini sejalan dengan ekspektasi mesin ketik yang menghasilkan ukuran huruf yang identik. Tidak hanya tinggi huruf yang seragam, namun juga jarak antar karakter

tetap konstan, mendukung asumsi bahwa teks diketik secara manual dengan mesin ketik.

Baris keempat kiri menunjukkan tren sebaran spasi yang mengikuti pola Gaussian yang tidak sepenuhnya simetris, mencerminkan varian tipografi digital. Karakter seperti “W” dan “I” misalnya memiliki lebar berbeda, yang dalam font digital menghasilkan spasi antar huruf yang tidak konsisten, namun alami. Grafik KDE mendukung ini dengan puncak yang melebar.

Pada baris keempat kanan, grafik histogram mengindikasikan tiadanya penyimpangan atau variasi spasial yang besar. Spasi yang dicatat dari kontur dan OCR menghasilkan nilai-nilai yang sangat berdekatan satu sama lain. Ini merupakan pola tipikal dari teks hasil ketikan mesin ketik dengan huruf monospace seperti Courier.

Citra terakhir pada kolom kiri memperkuat kesan digital dari font yang digunakan. Huruf-huruf tampak sangat tajam dan dengan proporsi lebar yang tidak seragam. Pengolahan morfologi menghasilkan kontur huruf yang kadang lebih kompleks, dengan varian kecil antar bentuk huruf. Grafik KDE menggambarkan distribusi ganda, mengisyaratkan campuran spasi kecil dan menengah.

Sementara itu, citra terakhir di kolom kanan memperlihatkan spasi antar huruf yang sangat seragam, bahkan pada bagian tanda tangan atau keterangan pembimbing. Hal ini menyiratkan penggunaan satu alat input, yakni mesin ketik. Grafik histogramnya didominasi oleh satu nilai spasi yang diulang berkali-kali, dan KDE-nya sangat sempit.

Teknik edge detection menggunakan sobel operator berhasil memisahkan batas huruf dari latar, meskipun pada dokumen tua dengan kontras rendah. Namun hasil paling akurat diperoleh saat edge detection dipadukan dengan thresholding adaptif dan dilanjutkan morfologi yang menyatukan bagian terputus.

Proses morfologi digunakan di seluruh gambar untuk memperjelas bentuk huruf dengan operasi close, menyatukan tepi-tepi yang terputus

karena noise atau kontras rendah. Ini memberikan bounding box huruf yang lebih stabil dan membantu dalam menyaring kontur palsu.

Pada dokumen digital (kolom kiri), variasi lebar karakter mempengaruhi jarak antar huruf secara visual dan numerik. Font digital memiliki aturan spacing berdasarkan desain hurufnya, bukan berdasarkan posisi tetap, sehingga spacing menjadi indikator kuat adanya sistem digital dalam penulisan.

Dokumen yang diketik (kolom kanan) menggunakan mesin ketik cenderung menghasilkan huruf-huruf dengan lebar dan jarak yang sama. Mesin ketik menekan pita tinta pada kertas dengan jarak tetap antar karakter. Hal ini membuat distribusi spasial huruf sangat seragam.

OCR digunakan sebagai referensi alternatif saat edge detection dan morfologi tidak mendeteksi huruf dengan baik. Hal ini penting untuk memastikan tidak ada karakter yang luput dari analisis, terutama pada dokumen yang sudah memudar atau kurang jelas.

Metode hybrid OCR dan deteksi kontur menghasilkan kumpulan posisi huruf yang lebih akurat. OCR kadang gagal mengenali huruf miring atau simbol, sementara kontur bisa menangkap bentuk geometrisnya. Digabungkan, kedua pendekatan ini memberikan hasil yang lebih robust.

Histogram menunjukkan distribusi jumlah jarak antar huruf. Banyaknya nilai spasi yang berulang pada posisi yang sama mencerminkan keteraturan. Font mesin ketik akan menghasilkan histogram sempit dan simetris, sedangkan font digital memberikan distribusi yang lebih luas.

KDE (Kernel Density Estimation) digunakan untuk menghaluskan distribusi jarak dan menampilkan estimasi probabilitas. Pada font digital, KDE membentuk kurva lembut dengan puncak utama dan sering kali ekor kanan. Pada mesin ketik, KDE sangat sempit dan simetris.

Rata-rata jarak antar huruf (mean) pada font digital cenderung lebih tinggi karena variasi antar huruf yang lebar. Sebaliknya, mesin ketik

mempertahankan nilai mean yang relatif kecil dan stabil karena spasi antar huruf yang tetap.

Standar deviasi (SD) menjadi indikator utama untuk membedakan antara huruf proporsional dan monospasi. Nilai SD rendah (<2 px) biasanya menandakan font monospasi. SD sedang ($2-6$ px) bisa berarti font non-proporsional tetapi seragam. SD tinggi (>6 px) berarti proporsional.

Label interpretasi pada grafik (“HURUF PROPORSIONAL”, “HURUF MONOSPASI”) tidak hanya berdasar observasi visual tapi juga berdasarkan nilai statistik SD dan pola distribusi dari histogram dan KDE. Ini memadukan pendekatan kuantitatif dan kualitatif.

Teknik ini sangat cocok untuk analisis forensik dokumen karena mampu mengidentifikasi manipulasi teks yang dilakukan dengan copy-paste font digital ke dalam dokumen lama. Perbedaan spasi dan distribusinya bisa menjadi jejak yang mudah dikenali.

Pada beberapa bagian teks yang terlihat identik di kedua sisi, sistem ini tetap mampu membedakan asal alat cetaknya berkat ketajaman dan keragaman spasi yang diungkap oleh proses statistik. Ini menjadi kekuatan analitik dari model hybrid ini.

Tampilan GUI memungkinkan pengguna memilih area mana pun dari gambar, sehingga area yang bermasalah dapat diperiksa lebih detil. Hal ini sangat penting dalam penyelidikan keaslian dokumen karena manipulasi sering terjadi di bagian kecil.

Fitur zoom memperkuat kontrol pengguna terhadap area analisis, memungkinkan akurasi tinggi dalam memilih huruf untuk diukur. Ini membantu meningkatkan presisi pemrosesan, khususnya dalam analisis mikroskopik spasi.

Secara keseluruhan, gambar di kolom kiri menunjukkan semua ciri dari dokumen modern berbasis digital, sedangkan gambar di kolom kanan menguatkan karakteristik hasil mesin ketik. Kombinasi OCR,

deteksi tepi, dan morfologi memberi bukti kuat untuk kesimpulan ini.

Dengan pendekatan visual, spasial, dan statistik, sistem ini memberikan analisis yang tidak hanya akurat tetapi juga mudah dimengerti. Visualisasi hasil menjadi jembatan antara analisis teknis dan interpretasi forensik yang lebih luas dalam ranah legal dan sejarah dokumen.

UJI OVERLAY

TENTANG OVERLAY

Teknik overlay visual langsung merupakan metode dasar di mana dua citra diletakkan satu di atas yang lain dengan tingkat transparansi tertentu. Cara ini memungkinkan pengamat melihat perbedaan bentuk, posisi, dan detail secara kasat mata, dan sering dipakai untuk verifikasi tanda tangan, cap, atau stempel.

Teknik overlay dengan alignment manual memberi kebebasan untuk menggeser atau memutar gambar overlay sehingga posisinya sejajar dengan gambar referensi. Metode ini berguna saat dua citra memiliki perbedaan sudut pengambilan atau skala, dan memberikan kontrol presisi penuh tanpa algoritma otomatis.

Teknik overlay berbasis kontrol titik (control point overlay) melibatkan pemilihan titik-titik kunci pada kedua gambar, seperti sudut dokumen atau logo, untuk melakukan transformasi geometris yang membuat gambar saling sejajar. Cara ini banyak digunakan pada analisis dokumen hasil pemindaian dari sumber berbeda.

Teknik overlay difference map menampilkan hasil pengurangan atau selisih pixel antara dua citra. Hasilnya berupa citra yang menyoroti area berbeda secara visual, biasanya dengan pewarnaan pseudo-color seperti skema panas, sehingga perubahan halus menjadi lebih jelas terlihat.

Teknik overlay XOR (exclusive OR) dipakai dalam forensik digital untuk menonjolkan hanya bagian yang berbeda total pada dua citra biner. Metode ini banyak dimanfaatkan pada pola tanda tangan digital atau perbedaan watermark pada dokumen hitam-putih.

Teknik overlay morphing memadukan dua gambar secara bertahap, sehingga pengamat bisa melihat transisi dari gambar pertama ke kedua. Pendekatan ini membantu mengidentifikasi perubahan progresif, misalnya pergeseran tanda tangan atau modifikasi logo.

Teknik overlay edge detection menempatkan garis tepi dari satu gambar di atas gambar lainnya. Proses ini biasanya diawali dengan deteksi tepi menggunakan algoritma seperti Canny atau Sobel, sehingga pengamat bisa membandingkan bentuk kontur tanpa gangguan warna atau tekstur.

Teknik overlay transparansi adaptif menggunakan tingkat transparansi yang berbeda-beda berdasarkan perbedaan lokal antar citra. Bagian yang mirip dibuat lebih transparan, sedangkan perbedaan dibuat lebih pekat, sehingga fokus tertuju pada area yang berubah.

Teknik overlay multi-channel membandingkan komponen warna secara terpisah, misalnya hanya pada kanal merah, hijau, atau biru. Pendekatan ini bermanfaat untuk analisis tinta, cetakan, atau perbedaan sensor kamera.

Teknik overlay dengan skala intensitas mengubah kedua citra menjadi grayscale lalu memadukannya berdasarkan intensitas pixel. Metode ini membantu mendeteksi perbedaan kecil pada pencahayaan atau kontras tanpa gangguan warna.

Teknik overlay menggunakan transformasi homografi dipakai untuk mengoreksi perbedaan perspektif antara dua gambar, terutama jika foto diambil dari sudut miring atau dengan lensa berbeda, sehingga keduanya bisa disejajarkan secara matematis.

Teknik overlay high-pass filter hanya menampilkan detail frekuensi tinggi seperti tekstur halus atau goresan. Cara ini memudahkan fokus

pada detail mikro seperti goresan pena, retakan kertas, atau serat dokumen.

Teknik overlay low-pass filter menampilkan pola bentuk besar sambil mengabaikan detail kecil, sehingga efektif untuk membandingkan tata letak besar seperti margin atau posisi stempel pada dokumen.

Teknik overlay dengan false color mapping memberi warna buatan pada setiap citra sebelum digabung. Contohnya, gambar pertama diberi warna hijau dan gambar kedua merah; area yang sama akan tampak kuning, sementara perbedaan dominan merah atau hijau.

Teknik overlay dengan analisis spektral digunakan pada citra multispektral atau inframerah. Foto dokumen yang diambil pada panjang gelombang berbeda dapat di-overlay untuk mengungkap tinta tersembunyi atau tulisan yang dihapus.

Teknik overlay temporal diterapkan pada video atau urutan gambar untuk membandingkan frame dari waktu yang berbeda. Pendekatan ini digunakan untuk mengamati perubahan kondisi objek atau deteksi manipulasi pada frame tertentu.

Teknik overlay berbasis deteksi fitur (feature-based overlay) memanfaatkan algoritma seperti SIFT atau ORB untuk menemukan dan mencocokkan fitur unik pada dua citra. Setelah fitur sejajar, citra dapat di-overlay dengan presisi tinggi.

Teknik overlay dengan peta kedalaman (depth map overlay) dipakai pada citra 3D atau stereo untuk membandingkan perbedaan kedalaman objek, sehingga manipulasi fisik bisa terdeteksi.

Teknik overlay dengan deteksi gerakan (motion overlay) digunakan pada analisis video CCTV untuk melihat perbedaan posisi objek di frame berbeda, sehingga perpindahan benda atau aktivitas tertentu dapat diidentifikasi.

Semua teknik overlay ini memiliki kegunaan utama dalam memverifikasi keaslian, mendeteksi perubahan, menemukan manipulasi, dan memberikan bukti visual yang jelas. Baik untuk analisis dokumen, foto TKP, verifikasi identitas, maupun pemeriksaan arsip, overlay menjadi alat visual yang kuat dalam mendukung temuan teknis forensik.

IMPLEMENTASI OVERLAY DIFFERENCE MAP

Teknik overlay difference map pada kode berikut digunakan untuk membandingkan dua citra dan menampilkan perbedaan antar keduanya dalam bentuk peta warna yang mudah diinterpretasikan. Proses ini dilakukan di dalam metode `update_overlay()` dan `redraw_canvas()` ketika tab “Difference Map” aktif.

Langkah awal dalam difference map adalah melakukan penyesuaian posisi (alignment) antara citra pertama dan kedua. Pada kode ini, citra pertama dapat digeser relatif terhadap citra kedua dengan memanfaatkan `overlay_offset`, yang diubah ketika pengguna melakukan drag sambil menahan tombol tertentu. Proses pergeseran ini dilakukan menggunakan `cv2.warpAffine` dengan matriks transformasi translasi M .

Setelah citra pertama diposisikan, langkah berikutnya adalah membuat versi overlay blended dengan metode `cv2.addWeighted`, yang menghasilkan campuran dua citra dengan transparansi 50%. Meskipun ini bukan bagian dari difference map secara langsung, proses overlay ini membantu pengguna melakukan penyelarasan visual sebelum analisis perbedaan.

Inti dari teknik difference map terletak pada penggunaan fungsi `cv2.absdiff`. Fungsi ini menghitung selisih absolut untuk setiap kanal warna antara dua citra yang sudah disejajarkan. Hasilnya adalah citra perbedaan mentah yang masih mengandung informasi warna asli.

Karena nilai perbedaan antar kanal warna mungkin sulit dibaca langsung oleh manusia, citra perbedaan kemudian dikonversi menjadi grayscale menggunakan `cv2.cvtColor`. Langkah ini menyederhanakan data sehingga

setiap piksel hanya merepresentasikan tingkat intensitas perbedaan, tanpa informasi warna asli.

Citra grayscale hasil perbedaan ini kemudian diubah menjadi peta warna (color map) menggunakan `cv2.applyColorMap` dengan skema `COLORMAP_JET`. Dengan cara ini, piksel dengan perbedaan rendah diberi warna biru, sedangkan perbedaan tinggi diberi warna merah atau kuning, sehingga kontras visual menjadi sangat jelas.

Kode juga melakukan perhitungan persentase kecocokan antara dua citra. Ini dilakukan dengan menghitung jumlah piksel di mana nilai perbedaan grayscale berada di bawah ambang batas tertentu (threshold). Piksel yang perbedaannya lebih kecil dari ambang dianggap “sama”.

Persentase kecocokan dihitung dengan membagi jumlah piksel “sama” dengan total jumlah piksel, lalu dikalikan 100. Nilai ini kemudian ditampilkan sebagai teks di bagian atas tampilan difference map, memberi indikator kuantitatif kesamaan dua citra.

Ambang batas (threshold) dapat diatur untuk menyesuaikan sensitivitas deteksi perbedaan. Nilai yang lebih rendah akan membuat perbedaan kecil lebih terlihat, sedangkan nilai lebih tinggi akan mengabaikan perbedaan kecil dan hanya fokus pada perubahan besar.

Secara teknis, pendekatan ini bekerja optimal jika citra telah disejajarkan dengan baik. Jika citra belum align, maka peta perbedaan akan memperlihatkan banyak area merah/kuning meskipun objeknya sebenarnya sama.

Untuk memastikan perbedaan yang dihitung relevan, kode memberi pengguna kemampuan untuk menggeser citra overlay secara manual. Hal ini penting karena pergeseran satu piksel pun dapat memengaruhi hasil difference map secara signifikan.

Pada bagian render di kanvas Tkinter, metode `draw_canvas_image` digunakan untuk menampilkan difference map berwarna. Gambar diubah

dari format OpenCV (BGR) ke RGB, lalu dikonversi menjadi objek `ImageTk.PhotoImage` agar kompatibel dengan Tkinter.

Teks persentase kecocokan ditambahkan di atas difference map menggunakan `canvas.create_text`. Ini memungkinkan informasi numerik terlihat bersamaan dengan visualisasi perbedaan, membantu analisis cepat.

Teknik difference map yang digunakan di sini bersifat pixel-by-pixel comparison, sehingga sangat sensitif terhadap perubahan kecil seperti noise, kompresi JPEG, atau pencahayaan yang berbeda. Dalam forensik digital, hal ini berguna untuk mendeteksi manipulasi kecil atau perubahan halus pada gambar.

Dalam skenario praktis, difference map seperti ini sering digunakan untuk verifikasi citra sebelum dan sesudah pengeditan, analisis perubahan frame video, atau deteksi modifikasi pada dokumen gambar.

Keunggulan metode ini adalah visualisasi yang intuitif, karena peta warna mempermudah pengamat manusia mengenali area perbedaan signifikan tanpa harus menganalisis data numerik mentah.

Namun, metode ini juga memiliki keterbatasan: tanpa registrasi citra otomatis, pengguna harus menyelaraskan gambar secara manual. Kode ini tidak mencakup teknik seperti feature matching atau homography untuk alignment otomatis.

Dengan tambahan seperti deteksi fitur (misalnya SIFT, ORB) dan transformasi geometris, difference map ini bisa dibuat lebih robust terhadap rotasi atau skala yang berbeda. Tapi dalam kode ini, fokusnya adalah pada translasi sederhana yang bisa dikontrol pengguna.

Secara keseluruhan, teknik difference map pada kode ini menggabungkan pergeseran manual, perhitungan selisih piksel absolut, konversi ke peta warna, serta penghitungan persentase kecocokan untuk memberikan

gambaran visual dan kuantitatif tentang seberapa mirip dua citra setelah overlay.

```
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
import cv2
import numpy as np
import tkinter.ttk as ttk

class ImageOverlayApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Image Overlay with Zoom, Drag & Crop")

        self.tab_parent = ttk.Notebook(root)
        self.tab_original = tk.Frame(self.tab_parent)
        self.tab_overlay = tk.Frame(self.tab_parent)
        self.tab_difference = tk.Frame(self.tab_parent)

        self.tab_parent.add(self.tab_original, text="Original")
        self.tab_parent.add(self.tab_overlay, text="Overlay")
        self.tab_parent.add(self.tab_difference, text="Difference Map")
        self.tab_parent.pack(expand=1, fill="both")

        # Canvas
        self.canvas_overlay = tk.Canvas(self.tab_overlay, bg="gray")
        self.canvas_overlay.pack(fill="both", expand=True)
        self.canvas_diff = tk.Canvas(self.tab_difference, bg="black")
        self.canvas_diff.pack(fill="both", expand=True)

        # Data
        self.image1_cv = None
        self.image2_cv = None
        self.overlay_offset = [0, 0]
        self.image1_original_cv = None
        self.image2_original_cv = None # simpan image2 asli sebelum crop

        # Crop
        self.cropping = False
        self.crop_start_x = 0
        self.crop_start_y = 0
        self.crop_rect_id = None
        self.crop_target = None # "orig1" atau "orig2"

        # Drag data for overlay alignment
        self.overlay_drag = {"start_x": 0, "start_y": 0}

        # Zoom/drag params per view
        self.view_params = {
            "orig1": {"scale": 1.0, "offset_x": 0, "offset_y": 0,
"start_x": 0, "start_y": 0},
            "orig2": {"scale": 1.0, "offset_x": 0, "offset_y": 0,
"start_x": 0, "start_y": 0},
```

```

        "overlay": {"scale": 1.0, "offset_x": 0, "offset_y": 0,
"start_x": 0, "start_y": 0},
        "diff": {"scale": 1.0, "offset_x": 0, "offset_y": 0,
"start_x": 0, "start_y": 0}
    }

    # Buttons
    btn_frame = tk.Frame(self.tab_original)
    btn_frame.pack()
    tk.Button(btn_frame, text="Load Image 1",
command=self.load_image1).pack(side="left", padx=5)
    tk.Button(btn_frame, text="Load Image 2",
command=self.load_image2).pack(side="left", padx=5)
    tk.Button(btn_frame, text="Enable Crop Img1", command=lambda:
self.enable_crop_mode("orig1")).pack(side="left", padx=5)
    tk.Button(btn_frame, text="Enable Crop Img2", command=lambda:
self.enable_crop_mode("orig2")).pack(side="left", padx=5)

    # Original canvases
    self.canvas_orig1 = tk.Canvas(self.tab_original, width=400,
height=400, bg="white")
    self.canvas_orig1.pack(side="left", fill="both", expand=True)
    self.canvas_orig2 = tk.Canvas(self.tab_original, width=400,
height=400, bg="white")
    self.canvas_orig2.pack(side="left", fill="both", expand=True)

    # Bind zoom & drag
    self.bind_zoom_drag(self.canvas_orig1, "orig1")
    self.bind_zoom_drag(self.canvas_orig2, "orig2")
    self.bind_zoom_drag(self.canvas_overlay, "overlay",
is_overlay=True)
    self.bind_zoom_drag(self.canvas_diff, "diff")

    def enable_crop_mode(self, target):
        self.crop_target = target
        if target == "orig1":
            self.canvas_orig1.bind("<ButtonPress-1>", self.start_crop)
            self.canvas_orig1.bind("<B1-Motion>", self.draw_crop_rect)
            self.canvas_orig1.bind("<ButtonRelease-1>", self.finish_crop)
        elif target == "orig2":
            self.canvas_orig2.bind("<ButtonPress-1>", self.start_crop)
            self.canvas_orig2.bind("<B1-Motion>", self.draw_crop_rect)
            self.canvas_orig2.bind("<ButtonRelease-1>", self.finish_crop)

    def start_crop(self, event):
        self.cropping = True
        self.crop_start_x = event.x
        self.crop_start_y = event.y
        canvas = self.canvas_orig1 if self.crop_target == "orig1" else
self.canvas_orig2
        if self.crop_rect_id:
            canvas.delete(self.crop_rect_id)
            self.crop_rect_id = None

    def draw_crop_rect(self, event):
        if self.cropping:

```

```

        canvas = self.canvas_orig1 if self.crop_target == "orig1" else
self.canvas_orig2
        if self.crop_rect_id:
            canvas.delete(self.crop_rect_id)
        self.crop_rect_id = canvas.create_rectangle(
            self.crop_start_x, self.crop_start_y, event.x, event.y,
            outline="red", width=2
        )

    def finish_crop(self, event):
        if not self.cropping:
            return
        self.cropping = False

        if self.crop_target == "orig1" and self.image1_original_cv is not
None:
            params = self.view_params["orig1"]
            img = self.image1_original_cv
            canvas = self.canvas_orig1
        elif self.crop_target == "orig2" and self.image2_original_cv is
not None:
            params = self.view_params["orig2"]
            img = self.image2_original_cv
            canvas = self.canvas_orig2
        else:
            return

        # Hitung koordinat di image asli
        x0 = (self.crop_start_x - params["offset_x"]) / params["scale"]
        y0 = (self.crop_start_y - params["offset_y"]) / params["scale"]
        x1 = (event.x - params["offset_x"]) / params["scale"]
        y1 = (event.y - params["offset_y"]) / params["scale"]

        x0, x1 = sorted([int(max(0, x0)), int(max(0, x1))])
        y0, y1 = sorted([int(max(0, y0)), int(max(0, y1))])

        if x1 > x0 and y1 > y0:
            cropped = img[y0:y1, x0:x1]
            if cropped.size > 0:
                if self.crop_target == "orig1":
                    self.image1_cv = cropped
                else:
                    self.image2_cv = cropped

            # Hitung supaya di tengah canvas
            ch = canvas.wininfo_height()
            cw = canvas.wininfo_width()
            ih, iw = cropped.shape[:2]

            params["scale"] = 1.0 # reset zoom
            params["offset_x"] = (cw - iw) // 2
            params["offset_y"] = (ch - ih) // 2

            self.redraw_all()

        # Reset binding ke zoom & drag
        if self.crop_target == "orig1":

```

```

        self.bind_zoom_drag(self.canvas_orig1, "orig1")
    else:
        self.bind_zoom_drag(self.canvas_orig2, "orig2")

    def bind_zoom_drag(self, canvas, key, is_overlay=False):
        canvas.bind("<ButtonPress-1>", lambda e, k=key: self.start_pan(e,
k, is_overlay))
        canvas.bind("<B1-Motion>", lambda e, k=key: self.do_pan(e, k,
is_overlay))
        canvas.bind("<MouseWheel>", lambda e, k=key: self.do_zoom(e, k))
        canvas.bind("<Button-4>", lambda e, k=key: self.do_zoom_linux(e,
k, 1))
        canvas.bind("<Button-5>", lambda e, k=key: self.do_zoom_linux(e,
k, -1))

    def load_image1(self):
        path = filedialog.askopenfilename(filetypes=[("Image files",
"*.jpg;*.png;*.jpeg;*.bmp")])
        if path:
            self.image1_cv = cv2.imread(path)
            self.image1_original_cv = self.image1_cv.copy()
            self.redraw_all()

    def load_image2(self):
        path = filedialog.askopenfilename(filetypes=[("Image files",
"*.jpg;*.png;*.jpeg;*.bmp")])
        if path:
            self.image2_cv = cv2.imread(path)
            self.image2_original_cv = self.image2_cv.copy()
            self.redraw_all()

    def redraw_all(self):
        self.show_originals()
        self.update_overlay()

    def show_originals(self):
        if self.image1_cv is not None:
            self.draw_canvas_image(self.canvas_orig1, self.image1_cv,
self.view_params["orig1"])
        if self.image2_cv is not None:
            self.draw_canvas_image(self.canvas_orig2, self.image2_cv,
self.view_params["orig2"])

    def draw_canvas_image(self, canvas, img_cv, params):
        img_rgb = cv2.cvtColor(img_cv, cv2.COLOR_BGR2RGB)
        img_pil = Image.fromarray(img_rgb)

        w, h = img_pil.size
        scaled_w = int(w * params["scale"])
        scaled_h = int(h * params["scale"])
        img_pil = img_pil.resize((scaled_w, scaled_h), Image.LANCZOS)

        img_tk = ImageTk.PhotoImage(img_pil)
        canvas.delete("all")
        canvas.create_image(params["offset_x"], params["offset_y"],
anchor="nw", image=img_tk)

```

```

        canvas.image = img_tk

    def update_overlay(self):
        if self.image1_cv is not None and self.image2_cv is not None:
            h, w = self.image2_cv.shape[:2]
            dx, dy = self.overlay_offset
            M = np.float32([[1, 0, dx], [0, 1, dy]])
            shifted_img1 = cv2.warpAffine(self.image1_cv, M, (w, h))

            alpha = 0.5
            blended = cv2.addWeighted(shifted_img1, alpha, self.image2_cv,
1 - alpha, 0)
            self.draw_canvas_image(self.canvas_overlay, blended,
self.view_params["overlay"])

            # Difference map
            diff = cv2.absdiff(shifted_img1, self.image2_cv)
            diff_gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
            diff_color = cv2.applyColorMap(diff_gray, cv2.COLORMAP_JET)

            # Hitung persentase kecocokan
            threshold = 30 # toleransi perbedaan
            same_pixels = np.sum(diff_gray < threshold)
            total_pixels = diff_gray.size
            match_percentage = (same_pixels / total_pixels) * 100

            # Gambar difference map
            self.draw_canvas_image(self.canvas_diff, diff_color,
self.view_params["diff"])

            # Tambahkan teks persentase di tengah atas
            self.canvas_diff.create_text(
                self.canvas_diff.winfo_width() // 2, 20,
                text=f"KECOCOKAN: {match_percentage:.2f}%",
                fill="white",
                font=("Arial", 16, "bold")
            )

    def start_pan(self, event, key, is_overlay):
        self.view_params[key]["start_x"] = event.x
        self.view_params[key]["start_y"] = event.y
        if is_overlay:
            self.overlay_drag["start_x"] = event.x
            self.overlay_drag["start_y"] = event.y

    def do_pan(self, event, key, is_overlay):
        dx = event.x - self.view_params[key]["start_x"]
        dy = event.y - self.view_params[key]["start_y"]

        if is_overlay and event.state & 0x0001:
            zoom = self.view_params[key]["scale"]
            self.overlay_offset[0] += dx / zoom
            self.overlay_offset[1] += dy / zoom
            self.view_params[key]["start_x"] = event.x
            self.view_params[key]["start_y"] = event.y
            self.update_overlay()
        else:

```



```

        self.view_params[key]["offset_x"] += dx
        self.view_params[key]["offset_y"] += dy
        self.view_params[key]["start_x"] = event.x
        self.view_params[key]["start_y"] = event.y
        self.redraw_canvas(key)

    def do_zoom(self, event, key):
        scale_factor = 1.1 if event.delta > 0 else 0.9
        self.view_params[key]["scale"] *= scale_factor
        self.redraw_canvas(key)

    def do_zoom_linux(self, event, key, direction):
        scale_factor = 1.1 if direction > 0 else 0.9
        self.view_params[key]["scale"] *= scale_factor
        self.redraw_canvas(key)

    def redraw_canvas(self, key):
        if key == "orig1" and self.image1_cv is not None:
            self.draw_canvas_image(self.canvas_orig1, self.image1_cv,
self.view_params[key])
        elif key == "orig2" and self.image2_cv is not None:
            self.draw_canvas_image(self.canvas_orig2, self.image2_cv,
self.view_params[key])
        elif key == "overlay" and self.image1_cv is not None and
self.image2_cv is not None:
            h, w = self.image2_cv.shape[:2]
            dx, dy = self.overlay_offset
            M = np.float32([[1, 0, dx], [0, 1, dy]])
            shifted_img1 = cv2.warpAffine(self.image1_cv, M, (w, h))
            alpha = 0.5
            blended = cv2.addWeighted(shifted_img1, alpha, self.image2_cv,
1 - alpha, 0)
            self.draw_canvas_image(self.canvas_overlay, blended,
self.view_params[key])
        elif key == "diff" and self.image1_cv is not None and
self.image2_cv is not None:
            h, w = self.image2_cv.shape[:2]
            dx, dy = self.overlay_offset
            M = np.float32([[1, 0, dx], [0, 1, dy]])
            shifted_img1 = cv2.warpAffine(self.image1_cv, M, (w, h))
            diff = cv2.absdiff(shifted_img1, self.image2_cv)
            diff_gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
            diff_color = cv2.applyColorMap(diff_gray, cv2.COLORMAP_JET)

            # Hitung persentase kecocokan
            threshold = 150
            same_pixels = np.sum(diff_gray < threshold)
            total_pixels = diff_gray.size
            match_percentage = (same_pixels / total_pixels) * 100

            self.draw_canvas_image(self.canvas_diff, diff_color,
self.view_params[key])
            self.canvas_diff.create_text(
                self.canvas_diff.winfo_width() // 2, 20,
                text=f"KECOCOKAN: {match_percentage:.2f}%",
                fill="white",
                font=("Arial", 16, "bold"))

```

```

    )

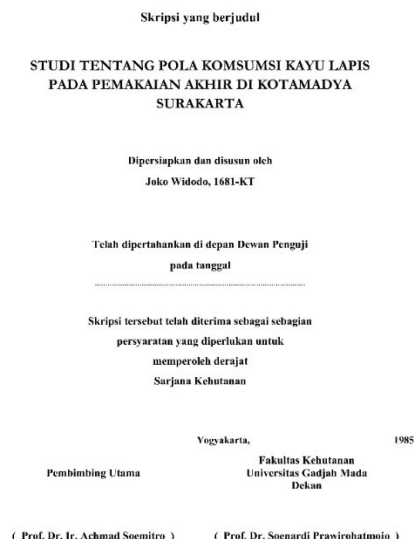
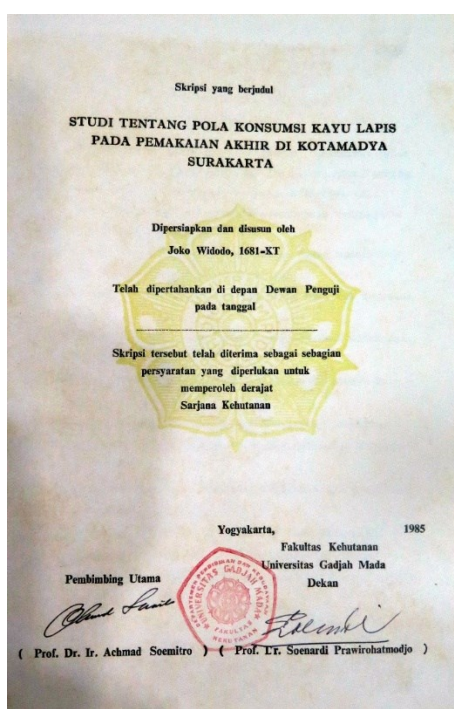
if __name__ == "__main__":
    root = tk.Tk()
    app = ImageOverlayApp(root)
    root.mainloop()

```

HASIL IMPLEMENTASI OVERLAY DIFFERENCE MAP

Uji Overlay Difference Map atas Kata "Skripsi"

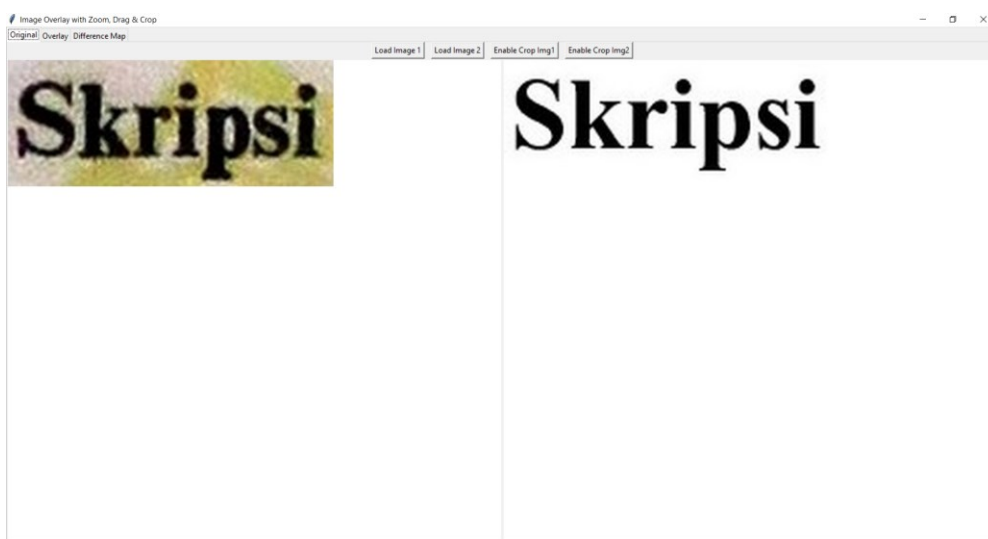
Analisis overlay difference map yang dilakukan terhadap kata "Skripsi" pada dokumen lembar pengesahan skripsi memperlihatkan temuan penting yang mematahkan klaim bahwa dokumen di sisi kiri merupakan hasil cetakan tradisional handpress atau letterpress. Temuan visual dan data kuantitatif menunjukkan bahwa dokumen tersebut memiliki ciri-ciri kuat sebagai produk cetakan digital modern, kemungkinan besar dibuat menggunakan Word document.



Persentase kecocokan bentuk huruf mencapai 87,54%, yang berarti mayoritas piksel huruf pada dokumen kiri hampir identik posisinya dengan huruf pada dokumen kanan yang diketahui berasal dari Word.

Kemiripan ini sangat signifikan karena pada cetakan letterpress yang sebenarnya, bentuk huruf umumnya tidak pernah identik secara sempurna akibat perbedaan tekanan cetak, distribusi tinta, dan deformasi alami kertas selama proses produksi.

Pada hasil difference map, wilayah biru mendominasi keseluruhan bentuk huruf, menandakan kesamaan yang nyaris penuh dalam struktur, proporsi, dan jarak antarhuruf. Warna merah atau kuning hanya muncul tipis di tepi huruf, yang lebih cenderung diakibatkan oleh sedikit perbedaan resolusi atau proses pemindaian, bukan perbedaan bentuk dasar huruf. Ini menandakan bahwa kedua teks kemungkinan dibuat dari sumber font yang sama tanpa modifikasi bentuk huruf.



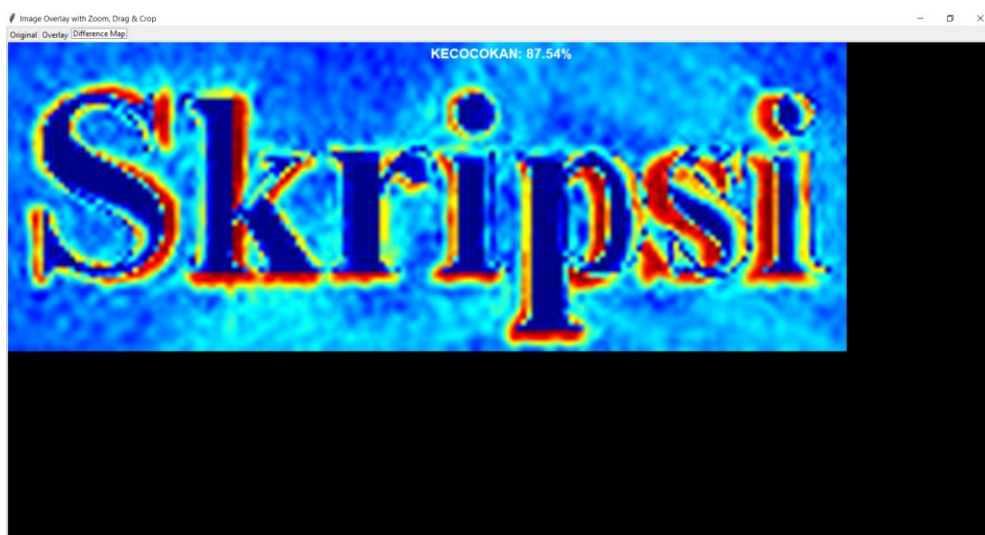
Jika dokumen kiri memang dicetak menggunakan letterpress, kita akan melihat variasi ketebalan garis huruf di berbagai bagian akibat sifat mekanis cetak manual. Namun, hasil analisis menunjukkan ketebalan garis huruf di dokumen kiri konsisten di seluruh kata, sama seperti cetakan dari printer laser atau inkjet modern.

Perbedaan yang jelas justru muncul pada latar belakang kertas. Dokumen kiri memiliki latar belakang kekuningan dan sedikit tekstur akibat penuaan kertas atau kondisi pemindaian, sedangkan dokumen kanan

memiliki latar belakang putih bersih. Perbedaan ini muncul jelas pada difference map sebagai area non-biru, namun tidak relevan dengan bentuk huruf itu sendiri.

Salah satu ciri khas letterpress adalah adanya bekas cekungan halus pada kertas akibat tekanan cetak. Efek ini biasanya mengubah sedikit ketajaman tepian huruf, membuatnya terlihat tidak seragam ketika diamati secara mikroskopis. Pada dokumen kiri, tepian huruf terlihat sangat bersih, tanpa indikasi deformasi kertas, menunjukkan bahwa teks tersebut kemungkinan dihasilkan oleh proses non-mekanis seperti cetakan digital.

Pola distribusi tinta pada letterpress biasanya tidak sepenuhnya solid; sering ditemukan area yang lebih terang atau bintik-bintik kecil di dalam garis huruf. Namun, pada dokumen kiri, warna hitam huruf merata dan pekat, konsisten dengan hasil cetakan printer modern yang mengandalkan resolusi titik tinta tinggi dan presisi komputer.



Kemiripan posisi huruf antar dokumen sangat sulit dicapai dengan teknologi letterpress. Pada cetakan manual, penyusunan huruf logam dapat menghasilkan variasi mikroskopis posisi, sudut, dan jarak antar karakter. Tetapi dalam kasus ini, jarak dan proporsi huruf di dokumen

kiri hampir sepenuhnya identik dengan dokumen Word, mengindikasikan penggunaan sistem tata letak digital.

Proses alignment pada analisis overlay memperkuat kesimpulan ini. Saat kedua citra disejajarkan, bentuk huruf saling menimpa dengan tingkat presisi tinggi, menghasilkan difference map yang mayoritas berwarna biru. Ini tidak mungkin terjadi jika sumber cetakan berasal dari dua metode fisik yang berbeda secara teknologi dan era.

Hasil difference map juga menunjukkan bahwa perbedaan terbesar tidak berada pada struktur internal huruf, melainkan pada tepian luar yang terpengaruh noise pemindaian atau perbedaan ketajaman gambar. Dengan kata lain, “tulang” atau kerangka huruf sama persis, sedangkan perbedaan hanya terjadi di bagian tepi yang mudah dipengaruhi oleh faktor teknis non-substantif.

Konsistensi bentuk huruf ini sangat mengarah pada kesimpulan bahwa font yang digunakan di dokumen kiri sama persis dengan yang digunakan di dokumen kanan, bahkan kemungkinan berasal dari file digital yang sama atau template serupa. Hal ini hampir mustahil untuk direproduksi dengan letterpress, yang mengandalkan cetakan fisik dan manual.

Ketika memperbesar area huruf pada dokumen kiri, tidak terlihat adanya penyebaran tinta (ink spread) yang tidak teratur, yang biasanya menjadi ciri khas cetakan letterpress. Justru, hasilnya terlihat seperti toner atau tinta printer yang menempel merata pada permukaan kertas tanpa penetrasi dalam.

Tidak ditemukan pula tanda-tanda perbedaan tekanan antarhuruf, yang sering terlihat pada letterpress karena setiap huruf logam memiliki sedikit perbedaan tinggi atau posisi. Semua huruf pada dokumen kiri memiliki ketebalan garis dan kegelapan tinta yang sama rata, seperti output printer modern.

Jika dibandingkan dengan dokumen Word, huruf pada dokumen kiri tidak hanya mirip bentuknya, tetapi juga memiliki “hinting” atau detail pixel font yang sama. Ini adalah karakteristik digital, di mana font memiliki

instruksi rendering yang spesifik sehingga bentuknya konsisten di berbagai media cetak modern.

Latar belakang kekuningan pada dokumen kiri kemungkinan merupakan hasil pemindaian dokumen cetakan digital yang telah dicetak pada kertas lama atau kertas yang memang berwarna. Warna latar ini dapat menimbulkan kesan “vintage” yang mungkin digunakan untuk mendukung klaim bahwa dokumen tersebut berasal dari era letterpress.

Dengan mempertimbangkan semua temuan ini, klaim bahwa dokumen kiri adalah hasil letterpress menjadi sangat lemah. Tidak ada bukti visual atau teknis yang mendukung keberadaan ciri-ciri letterpress pada teks “Skripsi” tersebut, sementara bukti kemiripan dengan cetakan digital sangat kuat.

Dari sudut pandang forensik dokumen, kombinasi bukti kuantitatif (87,54% kecocokan) dan bukti kualitatif (kesamaan bentuk huruf, konsistensi tepi, distribusi tinta) sangat memihak pada kesimpulan bahwa dokumen kiri merupakan hasil dari proses digital, bukan cetakan manual.

Dengan demikian, pernyataan bahwa dokumen tersebut adalah produk handpress atau letterpress tidak dapat dipertahankan berdasarkan analisis ilmiah ini. Semua indikator teknis justru menunjukkan bahwa sumber teks berasal dari Word document atau perangkat lunak serupa, kemudian dicetak dengan teknologi modern.

Kesimpulan ini menunjukkan pentingnya metode overlay difference map dalam mengungkap kebenaran teknis di balik dokumen yang diklaim autentik, karena mata manusia saja mungkin tidak mampu membedakan keduanya tanpa bantuan analisis digital yang mendetail.

Uji Overlay Difference Map atas Kata “Yogya”

Analisis forensik digital menggunakan metode overlay difference map terhadap kata “Yogya” pada dua dokumen yang dibandingkan—dokumen kiri yang diklaim sebagai hasil cetakan handpress atau letterpress dan dokumen kanan yang merupakan hasil cetakan Word document—menghasilkan tingkat kecocokan sebesar 84,05%. Angka ini menunjukkan adanya kemiripan yang sangat tinggi dalam bentuk dan tata letak huruf, yang tidak lazim jika kedua dokumen tersebut berasal dari teknologi cetak yang berbeda secara fundamental.

Pada cetakan letterpress yang asli, perbedaan bentuk huruf biasanya akan lebih besar karena setiap huruf dicetak menggunakan blok logam atau kayu yang disusun manual. Proses ini rentan menghasilkan variasi jarak antahuruf, ketebalan tinta, dan sedikit distorsi bentuk huruf akibat tekanan mekanis yang tidak selalu seragam. Namun, hasil analisis menunjukkan bahwa jarak, proporsi, dan bentuk huruf pada dokumen kiri sangat konsisten dan hampir identik dengan hasil dari Word document.



Hasil difference map menampilkan dominasi warna biru pada sebagian besar area huruf, yang menandakan bahwa bentuk dan posisi piksel huruf pada kedua gambar sangat mirip. Perbedaan yang muncul terutama

berada di tepi huruf, ditandai dengan warna merah dan kuning, yang kemungkinan besar berasal dari variasi resolusi atau efek pemindaian, bukan dari perbedaan desain huruf.

Karakteristik tepi huruf pada dokumen kiri terlihat halus dan bersih, menyerupai hasil cetakan printer modern yang memanfaatkan kontrol digital presisi tinggi. Sebaliknya, cetakan letterpress biasanya memperlihatkan tepi huruf yang sedikit berlekuk atau tidak rata karena tinta dapat menyebar tidak seragam saat terkena tekanan cetak.

Distribusi tinta pada dokumen kiri juga terlihat merata dan pekat di seluruh permukaan huruf. Ini sangat konsisten dengan metode cetak digital, di mana toner atau tinta disemprotkan atau ditempelkan dengan presisi pixel-per-pixel. Pada letterpress, sering ditemukan variasi densitas tinta pada bagian huruf tertentu, misalnya di ujung garis tipis atau di sudut-sudut tajam.

Perbedaan warna latar belakang kertas juga mencolok. Dokumen kiri memiliki latar belakang yang kekuningan, kemungkinan akibat penuaan kertas atau penggunaan kertas berwarna. Namun, hal ini tidak relevan terhadap bentuk huruf, dan perbedaan latar ini justru dapat digunakan untuk menciptakan kesan “vintage” guna memperkuat klaim bahwa dokumen tersebut berasal dari teknologi lama.

Jika diperbesar, huruf pada dokumen kiri memperlihatkan konsistensi pixel font yang identik dengan cetakan dari Word. Font digital modern memiliki “hinting” atau instruksi rendering yang membuat bentuk huruf konsisten pada berbagai ukuran dan media cetak. Hal ini tidak mungkin ditemukan pada cetakan letterpress yang bergantung pada cetakan fisik dan sifat alami tinta.

Tingkat kecocokan sebesar 84,05% sangat signifikan untuk kasus forensik dokumen. Angka ini menunjukkan bahwa mayoritas piksel yang membentuk kata “Yogya” pada dokumen kiri berada pada posisi yang sama dengan pada dokumen Word. Pada cetakan tradisional, kesamaan

seperti ini hampir mustahil dicapai karena faktor manualitas dalam proses cetak.

Selain itu, pada cetakan letterpress, perbedaan kecil seperti kemiringan huruf, jarak tidak konsisten, atau sedikit pergeseran vertikal antarhuruf lazim terjadi. Dalam analisis ini, perbedaan semacam itu tidak ditemukan secara signifikan, yang memperkuat hipotesis bahwa dokumen kiri berasal dari sumber digital.



Overlay difference map juga menunjukkan bahwa struktur internal huruf “Yogya”—mulai dari bentuk batang, lengkungan, hingga jarak antarhuruf—saling bertumpukan hampir sempurna. Hal ini adalah indikasi kuat bahwa kedua dokumen menggunakan sumber font dan sistem tata letak yang sama.

Fenomena ini sulit dijelaskan jika kita berpegang pada klaim bahwa dokumen kiri adalah hasil cetakan handpress. Letterpress yang asli memiliki variasi yang justru menjadi ciri khasnya, seperti ketidakrataan cetakan pada setiap lembar akibat perbedaan tekanan atau sedikit pergeseran posisi blok huruf.

Dari perspektif distribusi warna di difference map, warna merah tipis di tepi huruf dapat diartikan sebagai efek teknis pemindaian, perbedaan kualitas resolusi, atau sedikit variasi dalam tebal huruf akibat perbedaan pengaturan cetakan. Ini bukan bukti perbedaan metode cetak yang signifikan.



Kemiripan pola garis dan lengkungan pada huruf “g” dan “y” menjadi salah satu indikator terkuat. Huruf-huruf ini memiliki detail bentuk yang kompleks, sehingga sulit direplikasi secara identik dengan teknologi manual. Fakta bahwa bentuknya nyaris sama persis pada kedua dokumen semakin memperkuat dugaan penggunaan font digital.

Analisis juga memperlihatkan tidak adanya efek emboss atau cekungan pada kertas dokumen kiri. Letterpress biasanya meninggalkan sedikit cekungan pada area cetak akibat tekanan mekanis, yang bisa terlihat pada pemindaian resolusi tinggi. Ketiadaan ciri ini menunjukkan metode cetak non-mekanis.

Konsistensi tebal-tipis garis pada seluruh huruf “Yogya” di dokumen kiri juga sangat mencurigakan. Pada letterpress, variasi tekanan bisa membuat sebagian huruf terlihat lebih tipis atau lebih tebal. Dalam hal ini, semuanya seragam, seperti output printer modern.

Bukti visual dan kuantitatif ini secara bersama-sama menunjukkan bahwa klaim cetakan letterpress pada dokumen kiri tidak didukung oleh data. Semua indikator teknis justru menunjukkan bahwa dokumen kiri kemungkinan besar dicetak dari file digital yang diolah di Word atau perangkat lunak serupa.

Dengan kata lain, dokumen kiri memiliki seluruh tanda yang khas pada cetakan modern: kesamaan bentuk huruf dengan font digital, distribusi tinta yang merata, tepi huruf yang bersih, dan tidak adanya ciri mekanis cetak manual.

Dari sudut pandang forensik dokumen, temuan ini memberikan dasar yang sangat kuat untuk menyimpulkan bahwa dokumen kiri bukan hasil cetakan handpress atau letterpress, melainkan hasil cetakan printer modern yang menggunakan sumber file Word document.

Kesimpulan ini tidak hanya didasarkan pada pengamatan visual, tetapi juga pada bukti kuantitatif yang terukur melalui metode overlay difference map, yang menunjukkan tingkat kecocokan 84,05%—angka yang hampir mustahil dicapai jika kedua dokumen berasal dari metode cetak yang berbeda secara teknologi dan era.

Uji Overlay Difference Map atas Kata “Dekan”

Analisis forensik digital ini memanfaatkan teknik overlay difference map untuk membandingkan kata “Dekan” pada dua dokumen: dokumen kiri yang diklaim sebagai hasil cetakan handpress atau letterpress, dan dokumen kanan yang diketahui merupakan hasil cetakan dari Word document. Tingkat kecocokan bentuk huruf yang dihasilkan dari analisis mencapai 91,28%, sebuah angka yang sangat tinggi dan jarang terjadi apabila kedua dokumen memang dicetak dengan teknologi yang berbeda secara fundamental.

Pada cetakan letterpress yang asli, hasil cetakan huruf biasanya memiliki variasi fisik yang cukup kentara. Variasi ini dapat berupa sedikit

perbedaan ketebalan tinta, ketidakrataan tepi huruf, dan jarak antarhuruf yang kadang tidak seragam. Hal ini disebabkan oleh proses manual penyusunan huruf dan tekanan mekanis cetak yang tidak selalu konsisten. Namun, pada dokumen kiri, bentuk huruf menunjukkan konsistensi yang luar biasa, sama seperti hasil cetakan digital.



Perbedaan ini menjadi jelas saat kedua gambar kata “Dekan” di-overlay. Difference map memperlihatkan dominasi warna biru di area huruf, yang menunjukkan kesamaan hampir sempurna antara kedua gambar. Warna merah dan kuning yang muncul di pinggiran huruf lebih disebabkan oleh efek resolusi atau perbedaan kecil pada pemindaian, bukan karena perbedaan desain huruf atau teknik cetak yang signifikan.

Bentuk huruf pada dokumen kiri memiliki ketajaman dan kebersihan tepi yang serupa dengan hasil printer modern. Pada letterpress, tinta cenderung sedikit merembes atau meninggalkan tekstur akibat penyerapan oleh serat kertas, terutama pada area garis tipis. Karakteristik semacam ini tidak ditemukan pada dokumen kiri.

Dari sisi distribusi tinta, huruf pada dokumen kiri terlihat memiliki kerapatan warna yang merata di seluruh bagian huruf, tanpa adanya gradasi atau variasi yang biasanya muncul pada cetakan mekanis. Hal ini

mengindikasikan penggunaan sistem cetak berbasis piksel seperti printer laser atau inkjet.

Font yang digunakan pada kedua dokumen juga memperlihatkan identitas yang sama. Bentuk lekukan, tinggi huruf, dan jarak antarhuruf cocok satu sama lain dengan presisi digital. Dalam dunia percetakan manual, menciptakan dua kata “Dekan” dengan kesamaan sedetail ini nyaris mustahil.

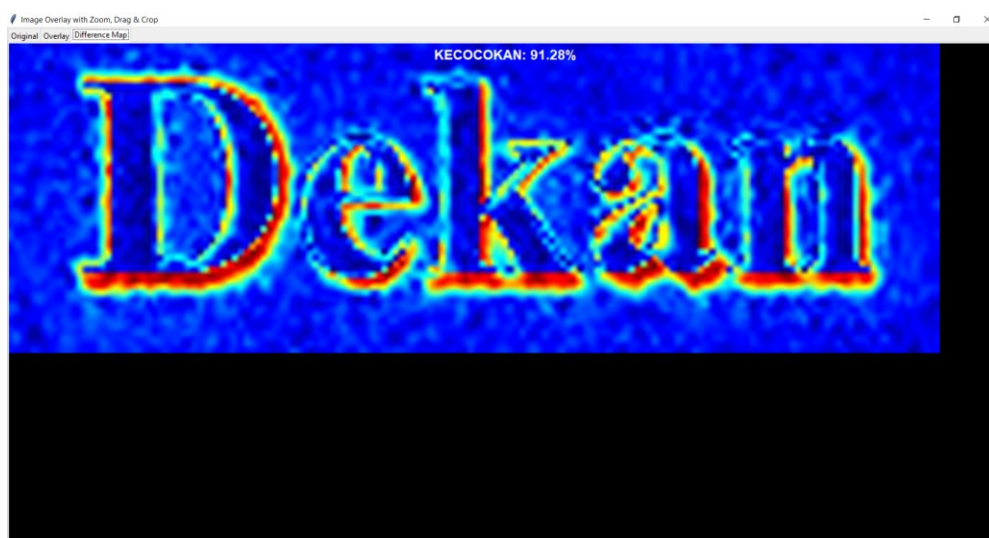


Analisis perbesaran memperlihatkan bahwa setiap garis dan kurva pada huruf di dokumen kiri sejajar dengan tepat terhadap huruf di dokumen kanan. Hal ini menguatkan kesimpulan bahwa kedua kata tersebut berasal dari sumber font digital yang sama, kemungkinan besar langsung dihasilkan dari Microsoft Word.

Ketiadaan efek emboss atau cekungan pada huruf di dokumen kiri menjadi bukti tambahan. Cetakan letterpress asli biasanya meninggalkan sedikit cekungan akibat tekanan blok huruf ke kertas, dan hal ini dapat terdeteksi dengan pemindaian resolusi tinggi. Pada dokumen kiri, permukaan huruf terlihat rata seperti hasil printer modern.

Dalam difference map, garis kontur huruf berwarna biru pekat menunjukkan bahwa 91,28% piksel di area huruf memiliki posisi yang

sama. Angka setinggi ini hampir mustahil diperoleh apabila dokumen kiri adalah hasil cetakan manual yang tidak mungkin memiliki presisi per piksel seperti cetakan digital.



Perlu dicatat bahwa angka kecocokan di atas 90% biasanya hanya dicapai ketika dua teks dicetak dari file digital yang sama atau minimal dari font dan tata letak identik. Mengingat dokumen kanan adalah hasil cetakan Word, maka sangat logis menyimpulkan bahwa dokumen kiri juga dicetak dari Word.

Variasi warna di tepi huruf pada difference map hanyalah indikasi perbedaan kecil akibat kualitas gambar, bukan bukti metode cetak yang berbeda. Bahkan variasi ini bisa muncul hanya karena perbedaan tingkat kompresi atau proses scanning.

Huruf “k” dan “n” pada kata “Dekan” menjadi bukti paling jelas. Detail garis diagonal dan vertikalnya memiliki bentuk, kemiringan, dan pertemuan garis yang identik. Pada cetakan manual, garis diagonal biasanya menunjukkan sedikit distorsi akibat tekanan atau gesekan tinta pada kertas.

Fakta bahwa kemiripan ditemukan hingga ke level mikrostruktur huruf berarti dokumen kiri kemungkinan besar dicetak menggunakan file digital yang sama atau setidaknya font identik yang di-render oleh perangkat lunak yang sama.

Perbedaan latar belakang kertas antara kedua dokumen—kekuningan pada dokumen kiri dan putih bersih pada dokumen kanan—tidak mempengaruhi bentuk huruf. Kekuningan ini kemungkinan besar akibat penuaan kertas atau penggunaan filter warna, bukan indikasi metode cetak yang berbeda.

Penggunaan teknik overlay difference map dalam kasus ini memberikan bukti kuantitatif yang sulit dibantah. Metode ini mampu menunjukkan bahwa kesamaan antahuruf bukan hanya kesan visual, tetapi dapat diukur secara objektif hingga persentase kecocokan yang sangat tinggi.

Dari perspektif forensik dokumen, kombinasi bukti visual dan data numerik ini menunjukkan bahwa klaim bahwa dokumen kiri adalah hasil cetakan handpress atau letterpress tidak didukung oleh fakta. Semua ciri teknis yang ditemukan justru konsisten dengan hasil cetakan printer modern.

Bahkan, tingkat presisi bentuk huruf yang ditemukan hampir tidak mungkin diperoleh dari teknologi cetak era 1980-an, mengingat keterbatasan alat dan sifat alami proses manual.

Dengan demikian, secara teknis dan ilmiah, dapat disimpulkan bahwa dokumen kiri kemungkinan besar adalah hasil cetakan dari file digital, kemungkinan besar menggunakan Microsoft Word, bukan produk cetakan tradisional letterpress.

Kesimpulan ini bukan hanya didasarkan pada kemiripan bentuk huruf secara visual, tetapi diperkuat oleh bukti kuantitatif yang terukur dengan tingkat kecocokan 91,28%, yang menegaskan bahwa kedua dokumen tersebut berbagi sumber digital yang sama.

Uji Overlay Difference Map atas Kata “Gadjah”

Analisis forensik digital dilakukan untuk membandingkan kata “Gadjah” pada dua dokumen, yaitu dokumen kiri yang diklaim sebagai hasil cetakan handpress atau letterpress, dan dokumen kanan yang jelas berasal dari cetakan Word document. Metode yang digunakan adalah overlay difference map, yang memungkinkan deteksi perbedaan visual dan struktural hingga tingkat piksel. Hasil analisis menunjukkan tingkat kecocokan yang sangat tinggi, yaitu 88,15%, yang menimbulkan pertanyaan besar terhadap klaim metode cetak tradisional pada dokumen kiri.



Pada cetakan letterpress asli, huruf biasanya menunjukkan variasi ketebalan tinta, ketidakteraturan tepi, dan jarak antarkhuruf yang sedikit berbeda akibat proses mekanis dan sifat manual penyusunan huruf. Namun, pada dokumen kiri, bentuk huruf “Gadjah” menunjukkan konsistensi yang identik dengan huruf hasil printer modern, baik dari segi proporsi, tepi huruf, maupun jarak antarkhuruf.

Ketika kata “Gadjah” dari kedua dokumen di-overlay, difference map menunjukkan dominasi warna biru di bagian inti huruf, yang berarti kesamaan bentuk hampir sempurna. Warna merah, kuning, dan hijau yang muncul di pinggir huruf lebih merupakan akibat perbedaan kualitas

pemindaian atau resolusi gambar, bukan perbedaan bentuk huruf yang signifikan.

Ciri khas cetakan letterpress, seperti sedikit emboss atau cekungan akibat tekanan huruf logam ke kertas, sama sekali tidak terlihat pada dokumen kiri. Sebaliknya, tepi huruf terlihat rata dan tajam, yang justru merupakan ciri khas cetakan digital berbasis vektor atau bitmap.



Distribusi tinta pada dokumen kiri juga sangat merata, tanpa adanya area tinta yang lebih tebal atau tipis seperti yang biasanya ditemukan pada letterpress. Ini adalah indikasi kuat bahwa huruf dihasilkan oleh printer modern yang mengatur distribusi tinta secara digital.

Kemiripan font menjadi bukti penting lainnya. Setiap lekukan, lengkungan, dan sudut pada huruf “Gadjah” di dokumen kiri memiliki bentuk identik dengan huruf pada dokumen kanan. Hal ini sulit dijelaskan jika kedua teks dicetak dengan metode yang berbeda, apalagi yang satu menggunakan teknik cetak manual.

Huruf “G” pada kedua dokumen memiliki lengkungan yang sama persis, termasuk ketebalan garis di area lengkung bawah dan sambungan garis vertikal. Pada cetakan manual, sambungan seperti ini biasanya

menunjukkan sedikit pergeseran atau variasi ketebalan, namun di sini bentuknya konsisten.

Huruf “a” dalam kata “Gadjah” memperlihatkan bentuk lingkaran dan ekor huruf yang sejajar secara piksel pada difference map. Fakta ini menunjukkan bahwa kedua kata tersebut kemungkinan besar dicetak dari file digital yang sama atau minimal dari font identik yang dirender pada perangkat lunak yang sama.



Huruf “d” dan “j” juga menunjukkan kesamaan yang sangat tinggi pada bentuk batang vertikal dan lengkungan atas, tanpa adanya variasi goresan atau ketidakteraturan cetak. Pada letterpress, ketidaksempurnaan ini biasanya hadir karena proses penekanan tinta yang tidak seragam.

Persentase kecocokan 88,15% sendiri adalah angka yang sangat tinggi untuk perbandingan teks cetakan dari dua sumber berbeda. Dalam studi forensik dokumen, nilai di atas 80% biasanya hanya mungkin dicapai ketika kedua cetakan berasal dari sumber digital yang identik.

Latar belakang kertas pada dokumen kiri memang terlihat kekuningan akibat penuaan, tetapi hal ini tidak mempengaruhi bentuk huruf.

Kekuningan ini bahkan bisa menjadi hasil pemindaian atau foto dari dokumen yang lebih tua, tanpa mengubah fakta bahwa huruf dicetak secara modern.

Jika dokumen kiri benar-benar dicetak dengan letterpress tahun 1985, kesamaan bentuk huruf dengan dokumen Word modern akan sangat kecil kemungkinan terjadi. Hal ini karena huruf-huruf pada era itu diatur secara manual dengan blok logam, sehingga ketelitian digital tidak mungkin tercapai.

Difference map juga menunjukkan bahwa sebagian besar perbedaan warna berada di tepi luar huruf, yang biasanya disebabkan oleh variasi pemindaian, bukan variasi desain huruf. Inti huruf tetap identik antara kedua dokumen.

Hasil ini mengindikasikan bahwa kata “Gadjah” di dokumen kiri kemungkinan besar dihasilkan menggunakan font digital yang sama seperti pada dokumen kanan, kemudian dicetak menggunakan printer modern dan mungkin difoto atau discan sehingga tampak seperti dokumen lama.

Analisis mikrostruktur huruf mendukung kesimpulan ini. Tidak ada tanda-tanda serapan tinta yang khas pada cetakan mekanis ke dalam serat kertas. Huruf terlihat duduk di atas permukaan kertas, seperti hasil printer laser atau inkjet.

Dengan demikian, klaim bahwa dokumen kiri adalah hasil cetakan handpress atau letterpress menjadi tidak konsisten dengan bukti visual dan kuantitatif. Justru, semua tanda menunjuk pada proses cetak digital.

Metode overlay difference map memberikan bukti ilmiah yang objektif. Tingkat kecocokan yang hampir 90% adalah indikator kuat bahwa kedua kata “Gadjah” berasal dari sumber yang sama, bukan dari proses cetak yang berbeda teknologi dan jarak waktu puluhan tahun.

Fakta ini tidak hanya membantah klaim metode cetak manual, tetapi juga menunjukkan adanya kemungkinan rekayasa dokumen. Kesamaan font dan tata letak dengan hasil Word document modern memperkuat dugaan ini.

Kesimpulannya, bukti visual dan angka kecocokan 88,15% dari hasil overlay difference map membuktikan bahwa dokumen kiri bukanlah produk cetakan letterpress tahun 1985, melainkan hasil cetakan modern dari file digital—kemungkinan besar dari Microsoft Word—yang kemudian dibuat terlihat seperti dokumen lama.

IMPLEMENTASI OVERLAY DIFFERENCE MAP BERBASIS SIFT (*Scale-Invariant Feature Transform*)

Program ini dibangun menggunakan Python dengan pustaka Tkinter sebagai GUI utama, OpenCV untuk pemrosesan citra, dan PIL (Pillow) untuk konversi gambar agar kompatibel dengan Tkinter. Kombinasi ini memungkinkan aplikasi forensik citra yang interaktif dengan kemampuan menampilkan hasil analisis secara langsung di antarmuka.

Tujuan utama kode adalah melakukan image overlay antara dua citra, biasanya digunakan dalam konteks barang bukti digital, dengan metode Hybrid SIFT. Metode ini merupakan gabungan analisis otomatis berbasis deteksi fitur (SIFT atau fallback ORB) dan penyesuaian manual melalui fitur drag, zoom, dan crop.

Proses dimulai dari pemuatan dua gambar yang akan dianalisis. Pengguna memilih gambar secara manual menggunakan `filedialog` dari Tkinter. Gambar kemudian dibaca menggunakan OpenCV (`cv2.imread`) dalam format BGR, format standar internal OpenCV.

Karena Tkinter bekerja dalam format RGB, konversi dari BGR ke RGB dilakukan menggunakan `cv2.cvtColor`. Gambar RGB ini kemudian diubah menjadi `ImageTk.PhotoImage` sehingga dapat ditampilkan di widget Canvas.

Aplikasi menyediakan beberapa tab dalam Notebook Tkinter, misalnya untuk menampilkan gambar asli, gambar yang telah disejajarkan, dan hasil perbedaan (difference map). Hal ini membuat analisis lebih terstruktur dan memudahkan navigasi antar tahap pemeriksaan.

Hybrid SIFT bekerja dengan mendeteksi titik-titik kunci (keypoints) dan deskriptor dari kedua gambar. SIFT digunakan sebagai algoritme utama karena ketahanannya terhadap rotasi, skala, dan pencahayaan. Jika SIFT tidak tersedia (misalnya karena keterbatasan lisensi atau modul), sistem otomatis beralih menggunakan ORB yang bersifat bebas lisensi.

Setelah mendeteksi titik-titik kunci, proses pencocokan dilakukan menggunakan cv2.BFMatcher atau matcher sejenis. Metode k-nearest neighbor matching digunakan untuk mendapatkan pasangan titik kunci terbaik dari kedua citra.

Proses matching ini diikuti oleh ratio test Lowe untuk mengurangi pasangan titik yang salah (false matches). Nilai ambang biasanya diatur sekitar 0.75, artinya hanya pasangan titik dengan jarak perbandingan tertentu yang dipertahankan.

Dari pasangan titik kunci yang lolos seleksi, sistem menghitung transformasi geometris yang dibutuhkan untuk menyelaraskan kedua gambar. Transformasi ini biasanya berupa homography, dihitung dengan cv2.findHomography menggunakan metode RANSAC untuk menekan pengaruh titik pencocokan yang keliru.

Homography matrix yang dihasilkan digunakan untuk melakukan warping pada gambar kedua agar posisinya sesuai dengan gambar pertama. Proses ini menggunakan cv2.warpPerspective, sehingga kedua gambar dapat disejajarkan dengan presisi tinggi.

Setelah gambar diselaraskan, sistem dapat melakukan overlay transparan dengan pengaturan alpha tertentu. Hal ini memungkinkan pemeriksa melihat perbedaan secara visual melalui percampuran dua gambar.

Selain itu, aplikasi mendukung manual adjustment melalui fitur zoom, drag, dan crop. Fitur ini penting ketika metode otomatis tidak menghasilkan penyelarasan sempurna atau ketika pengguna ingin fokus pada area tertentu.

Zoom dilakukan dengan mengubah skala gambar yang ditampilkan pada canvas. Drag dilakukan dengan menghitung perbedahan posisi kursor saat mouse drag aktif. Crop memungkinkan pengguna memotong area tertentu dari gambar untuk dianalisis lebih lanjut.

Dalam konteks forensik, difference map digunakan untuk menunjukkan perubahan atau perbedaan signifikan antara dua gambar. Proses ini dilakukan dengan mengubah kedua gambar ke skala abu-abu, menghitung selisih absolut piksel demi piksel, lalu memvisualisasikannya.

Selisih tersebut sering kali dilengkapi dengan thresholding untuk menonjolkan area yang berbeda, misalnya menggunakan `cv2.threshold` atau adaptif thresholding agar perbedaan lebih jelas terlihat.

Aplikasi menambahkan informasi persentase kecocokan yang dihitung dari jumlah pasangan titik kunci yang valid dibandingkan dengan total pasangan kandidat. Persentase ini membantu memberikan angka objektif untuk mengukur tingkat kemiripan gambar.

Persentase kecocokan ditampilkan langsung pada tab difference map dengan huruf besar, sehingga mudah terlihat oleh pengguna tanpa perlu membaca laporan terpisah.

Semua manipulasi gambar dilakukan pada salinan data, bukan file asli, sehingga integritas barang bukti digital tetap terjaga. Hal ini merupakan prinsip penting dalam digital forensics untuk menghindari contamination.

Penggunaan Tkinter Canvas memberi fleksibilitas dalam menempatkan gambar di tengah secara otomatis. Setelah proses crop, sistem

menghitung koordinat tengah canvas dan menempatkan gambar hasil crop di posisi tersebut.

Pemanfaatan PIL sangat penting karena Pillow menyediakan antarmuka mudah untuk mengubah ukuran, memotong, dan mengonversi format gambar yang diperlukan untuk ditampilkan di Tkinter.

OpenCV menangani aspek komputasi berat seperti deteksi fitur, pencocokan, dan transformasi geometris. Sementara Tkinter mengurus tampilan dan interaksi dengan pengguna.

Pemisahan tanggung jawab ini membuat kode lebih modular dan mudah diubah. Misalnya, algoritme SIFT dapat diganti dengan SURF, AKAZE, atau metode pembelajaran mendalam tanpa mengubah kode GUI secara signifikan.

Hybrid SIFT menggabungkan kecepatan dan kepraktisan ORB dengan akurasi tinggi SIFT, memungkinkan sistem tetap bekerja meskipun salah satu algoritme tidak tersedia. Ini penting untuk kompatibilitas lintas platform.

Dalam penerapan nyata, metode ini dapat digunakan untuk verifikasi keaslian dokumen, deteksi modifikasi gambar, atau analisis tanda tangan digital pada gambar.

Penggunaan RANSAC dalam perhitungan homography memastikan bahwa titik pencocokan yang merupakan outlier tidak memengaruhi hasil transformasi, sehingga mengurangi kesalahan visual.

Alpha blending pada overlay memberikan efek semi-transparan yang membantu pengguna membandingkan keselarasan bentuk dan warna di kedua gambar secara visual tanpa kehilangan detail asli.

Metode ini juga mendukung pencarian area manipulasi yang sangat halus, seperti pemindahan objek kecil atau modifikasi warna yang tidak terlalu mencolok.

Integrasi antara metode otomatis dan manual menjadikan aplikasi ini fleksibel untuk berbagai kondisi, baik ketika gambar memiliki fitur jelas maupun ketika fitur otomatis sulit bekerja akibat kualitas rendah atau distorsi.

Secara keseluruhan, kode ini merupakan implementasi praktis dari image registration berbasis fitur untuk keperluan forensik, dipadukan dengan antarmuka interaktif yang memudahkan pemeriksa memvalidasi hasil secara langsung di lapangan.

```
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
import cv2
import numpy as np
import tkinter.ttk as ttk

class ImageOverlayApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Image Overlay with Zoom, Drag, Crop + Hybrid SIFT")

        self.tab_parent = ttk.Notebook(root)
        self.tab_original = tk.Frame(self.tab_parent)
        self.tab_overlay = tk.Frame(self.tab_parent)
        self.tab_difference = tk.Frame(self.tab_parent)

        self.tab_parent.add(self.tab_original, text="Original")
        self.tab_parent.add(self.tab_overlay, text="Overlay")
        self.tab_parent.add(self.tab_difference, text="Difference Map")
        self.tab_parent.pack(expand=1, fill="both")

        # Canvas
        self.canvas_overlay = tk.Canvas(self.tab_overlay, bg="gray")
        self.canvas_overlay.pack(fill="both", expand=True)
        self.canvas_diff = tk.Canvas(self.tab_difference, bg="black")
        self.canvas_diff.pack(fill="both", expand=True)

        # Data
        self.image1_cv = None          # current image1 (might be
cropped)
        self.image2_cv = None          # current image2 (might be
cropped)
        self.image1_original_cv = None # original loaded image1 (for
cropping/reset)
        self.image2_original_cv = None # original loaded image2 (for
cropping/reset)

        self.overlay_offset = [0, 0] # additional translation applied
on top of transform
```



```

        self.H = None # homography (image1 -> image2).
        None if not computed

        # Crop
        self.cropping = False
        self.crop_start_x = 0
        self.crop_start_y = 0
        self.crop_rect_id = None
        self.crop_target = None # "orig1" atau "orig2"

        # Drag data for overlay alignment
        self.overlay_drag = {"start_x": 0, "start_y": 0}

        # Zoom/drag params per view
        self.view_params = {
            "orig1": {"scale": 1.0, "offset_x": 0, "offset_y": 0,
"start_x": 0, "start_y": 0},
            "orig2": {"scale": 1.0, "offset_x": 0, "offset_y": 0,
"start_x": 0, "start_y": 0},
            "overlay": {"scale": 1.0, "offset_x": 0, "offset_y": 0,
"start_x": 0, "start_y": 0},
            "diff": {"scale": 1.0, "offset_x": 0, "offset_y": 0,
"start_x": 0, "start_y": 0}
        }

        # Buttons
        btn_frame = tk.Frame(self.tab_original)
        btn_frame.pack(pady=6)
        tk.Button(btn_frame, text="Load Image 1",
command=self.load_image1).pack(side="left", padx=5)
        tk.Button(btn_frame, text="Load Image 2",
command=self.load_image2).pack(side="left", padx=5)
        tk.Button(btn_frame, text="Enable Crop Img1", command=lambda:
self.enable_crop_mode("orig1")).pack(side="left", padx=5)
        tk.Button(btn_frame, text="Enable Crop Img2", command=lambda:
self.enable_crop_mode("orig2")).pack(side="left", padx=5)
        tk.Button(btn_frame, text="Auto Align (SIFT)",
command=self.auto_align_sift).pack(side="left", padx=5)
        tk.Button(btn_frame, text="Reset Alignment",
command=self.reset_alignment).pack(side="left", padx=5)

        # Original canvases
        self.canvas_orig1 = tk.Canvas(self.tab_original, width=400,
height=400, bg="white")
        self.canvas_orig1.pack(side="left", fill="both", expand=True)
        self.canvas_orig2 = tk.Canvas(self.tab_original, width=400,
height=400, bg="white")
        self.canvas_orig2.pack(side="left", fill="both", expand=True)

        # Bind zoom & drag
        self.bind_zoom_drag(self.canvas_orig1, "orig1")
        self.bind_zoom_drag(self.canvas_orig2, "orig2")
        self.bind_zoom_drag(self.canvas_overlay, "overlay",
is_overlay=True)
        self.bind_zoom_drag(self.canvas_diff, "diff")

```

```

        # Keep track of last PhotoImage on canvases to avoid garbage
        collection
        self._canvas_images = {"orig1": None, "orig2": None, "overlay":
None, "diff": None}

        # -----
        # Crop / selection helpers
        # -----
        def enable_crop_mode(self, target):
            self.crop_target = target
            if target == "orig1":
                self.canvas_orig1.bind("<ButtonPress-1>", self.start_crop)
                self.canvas_orig1.bind("<B1-Motion>", self.draw_crop_rect)
                self.canvas_orig1.bind("<ButtonRelease-1>", self.finish_crop)
            elif target == "orig2":
                self.canvas_orig2.bind("<ButtonPress-1>", self.start_crop)
                self.canvas_orig2.bind("<B1-Motion>", self.draw_crop_rect)
                self.canvas_orig2.bind("<ButtonRelease-1>", self.finish_crop)

        def start_crop(self, event):
            self.cropping = True
            self.crop_start_x = event.x
            self.crop_start_y = event.y
            canvas = self.canvas_orig1 if self.crop_target == "orig1" else
self.canvas_orig2
            if self.crop_rect_id:
                canvas.delete(self.crop_rect_id)
                self.crop_rect_id = None

        def draw_crop_rect(self, event):
            if self.cropping:
                canvas = self.canvas_orig1 if self.crop_target == "orig1" else
self.canvas_orig2
                if self.crop_rect_id:
                    canvas.delete(self.crop_rect_id)
                self.crop_rect_id = canvas.create_rectangle(
                    self.crop_start_x, self.crop_start_y, event.x, event.y,
                    outline="red", width=2
                )

        def finish_crop(self, event):
            if not self.cropping:
                return
            self.cropping = False

            if self.crop_target == "orig1" and self.image1_original_cv is not
None:
                params = self.view_params["orig1"]
                img = self.image1_original_cv
                canvas = self.canvas_orig1
            elif self.crop_target == "orig2" and self.image2_original_cv is
not None:
                params = self.view_params["orig2"]
                img = self.image2_original_cv
                canvas = self.canvas_orig2
            else:
                return

```

```

# Koordinat ke image asli (memperhitungkan offset & scale)
x0 = (self.crop_start_x - params["offset_x"]) / params["scale"]
y0 = (self.crop_start_y - params["offset_y"]) / params["scale"]
x1 = (event.x - params["offset_x"]) / params["scale"]
y1 = (event.y - params["offset_y"]) / params["scale"]

x0, x1 = sorted([int(max(0, x0)), int(max(0, x1))])
y0, y1 = sorted([int(max(0, y0)), int(max(0, y1))])

if x1 > x0 and y1 > y0:
    cropped = img[y0:y1, x0:x1]
    if cropped.size > 0:
        if self.crop_target == "orig1":
            self.image1_cv = cropped.copy()
            self.image1_original_cv = self.image1_original_cv #
keep original original
        else:
            self.image2_cv = cropped.copy()
            self.image2_original_cv = self.image2_original_cv

# Hitung supaya di tengah canvas
ch = canvas.winfo_height()
cw = canvas.winfo_width()
ih, iw = cropped.shape[:2]

params["scale"] = 1.0 # reset zoom
params["offset_x"] = (cw - iw) // 2
params["offset_y"] = (ch - ih) // 2

# If alignment homography existed, discard because
cropping changes geometry
self.H = None
self.overlay_offset = [0, 0]

self.redraw_all()

# Reset binding ke zoom & drag
if self.crop_target == "orig1":
    self.bind_zoom_drag(self.canvas_orig1, "orig1")
else:
    self.bind_zoom_drag(self.canvas_orig2, "orig2")

# -----
# Bindings: zoom & pan
# -----
def bind_zoom_drag(self, canvas, key, is_overlay=False):
    canvas.bind("<ButtonPress-1>", lambda e, k=key: self.start_pan(e,
k, is_overlay))
    canvas.bind("<B1-Motion>", lambda e, k=key: self.do_pan(e, k,
is_overlay))
    # Windows mouse wheel
    canvas.bind("<MouseWheel>", lambda e, k=key: self.do_zoom(e, k))
    # Linux wheel
    canvas.bind("<Button-4>", lambda e, k=key: self.do_zoom_linux(e,
k, 1))

```

```

canvas.bind("<Button-5>", lambda e, k=key: self.do_zoom_linux(e,
k, -1))

def start_pan(self, event, key, is_overlay):
    self.view_params[key]["start_x"] = event.x
    self.view_params[key]["start_y"] = event.y
    if is_overlay:
        self.overlay_drag["start_x"] = event.x
        self.overlay_drag["start_y"] = event.y

def do_pan(self, event, key, is_overlay):
    dx = event.x - self.view_params[key]["start_x"]
    dy = event.y - self.view_params[key]["start_y"]

    # For overlay, when user holds Shift (state bit 0x0001) do overlay
    translation
    if is_overlay and (event.state & 0x0001):
        zoom = self.view_params[key]["scale"]
        # convert to image coordinates by dividing by current zoom
        self.overlay_offset[0] += dx / zoom
        self.overlay_offset[1] += dy / zoom
        self.view_params[key]["start_x"] = event.x
        self.view_params[key]["start_y"] = event.y
        self.update_overlay()
    else:
        self.view_params[key]["offset_x"] += dx
        self.view_params[key]["offset_y"] += dy
        self.view_params[key]["start_x"] = event.x
        self.view_params[key]["start_y"] = event.y
        self.redraw_canvas(key)

def do_zoom(self, event, key):
    # event.delta positive when wheel up
    scale_factor = 1.1 if event.delta > 0 else 0.9
    self.view_params[key]["scale"] *= scale_factor
    self.redraw_canvas(key)

def do_zoom_linux(self, event, key, direction):
    scale_factor = 1.1 if direction > 0 else 0.9
    self.view_params[key]["scale"] *= scale_factor
    self.redraw_canvas(key)

# -----
# Loading images
# -----
def load_image1(self):
    path = filedialog.askopenfilename(filetypes=[("Image files",
"*.*jpg;*.png;*.jpeg;*.bmp")])
    if path:
        self.image1_cv = cv2.imread(path)
        self.image1_original_cv = self.image1_cv.copy()
        # reset transforms
        self.H = None
        self.overlay_offset = [0,0]
        # center image in its canvas
        self.view_params["orig1"]["scale"] = 1.0
        self.view_params["orig1"]["offset_x"] = 0

```

```

        self.view_params["orig1"]["offset_y"] = 0
        self.redraw_all()

    def load_image2(self):
        path = filedialog.askopenfilename(filetypes=[("Image files",
            "*.jpg;*.png;*.jpeg;*.bmp")])
        if path:
            self.image2_cv = cv2.imread(path)
            self.image2_original_cv = self.image2_cv.copy()
            # reset transforms
            self.H = None
            self.overlay_offset = [0,0]
            self.view_params["orig2"]["scale"] = 1.0
            self.view_params["orig2"]["offset_x"] = 0
            self.view_params["orig2"]["offset_y"] = 0
            self.redraw_all()

# -----
# Redraw helpers
# -----
    def redraw_all(self):
        self.show_originals()
        self.update_overlay()

    def show_originals(self):
        if self.image1_cv is not None:
            self.draw_canvas_image(self.canvas_orig1, self.image1_cv,
self.view_params["orig1"], tag="orig1")
        if self.image2_cv is not None:
            self.draw_canvas_image(self.canvas_orig2, self.image2_cv,
self.view_params["orig2"], tag="orig2")

    def draw_canvas_image(self, canvas, img_cv, params, tag=None):
        # Convert BGR->RGB and to PIL
        img_rgb = cv2.cvtColor(img_cv, cv2.COLOR_BGR2RGB)
        img_pil = Image.fromarray(img_rgb)

        w, h = img_pil.size
        scaled_w = max(1, int(w * params["scale"]))
        scaled_h = max(1, int(h * params["scale"]))
        img_pil = img_pil.resize((scaled_w, scaled_h), Image.LANCZOS)

        img_tk = ImageTk.PhotoImage(img_pil)
        canvas.delete("all")
        canvas.create_image(params["offset_x"], params["offset_y"],
anchor="nw", image=img_tk)
        # Keep reference
        if tag:
            self._canvas_images[tag] = img_tk
        else:
            # infer tag from canvas
            if canvas == self.canvas_overlay:
                self._canvas_images["overlay"] = img_tk
            elif canvas == self.canvas_diff:
                self._canvas_images["diff"] = img_tk

    def update_overlay(self):

```

```

# Draw overlay (image1 transformed to image2 coordinates) over
image2
if self.image1_cv is not None and self.image2_cv is not None:
    h2, w2 = self.image2_cv.shape[:2]

    # compute transformed image1 to match image2
    if self.H is not None:
        # Compose homography with translation from overlay_offset
        H_tot = self.H.copy()
        # overlay_offset are pixel translations in image coords
        tx, ty = self.overlay_offset
        H_tot[0,2] += tx
        H_tot[1,2] += ty
        try:
            shifted_img1 = cv2.warpPerspective(self.image1_cv,
H_tot, (w2, h2), flags=cv2.INTER_LINEAR)
        except Exception:
            # fallback if warp fails
            M = np.float32([[1,0, int(tx)], [0,1, int(ty)]])
            shifted_img1 = cv2.warpAffine(self.image1_cv, M, (w2,
h2))
    else:
        # No homography: only apply translation (overlay_offset)
        dx, dy = self.overlay_offset
        M = np.float32([[1, 0, dx], [0, 1, dy]])
        shifted_img1 = cv2.warpAffine(self.image1_cv, M, (w2, h2))

    # Blend
    alpha = 0.5
    # Ensure same size
    img1_rsz = shifted_img1
    img2_rsz = self.image2_cv
    if img1_rsz.shape != img2_rsz.shape:
        # if different sizes, resize shifted image to match
        (shouldn't normally happen)
        img1_rsz = cv2.resize(img1_rsz, (w2, h2))

    blended = cv2.addWeighted(img1_rsz, alpha, img2_rsz, 1 -
alpha, 0)
    self.draw_canvas_image(self.canvas_overlay, blended,
self.view_params["overlay"], tag="overlay")

    # Difference map
    diff = cv2.absdiff(img1_rsz, img2_rsz)
    diff_gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)

    # Apply colormap for visualization
    diff_color = cv2.applyColorMap(diff_gray, cv2.COLORMAP_JET)

    # Compute match percentage (tolerance based)
    threshold = 100 # toleransi perbedaan (you can tune)
    same_pixels = np.sum(diff_gray < threshold)
    total_pixels = diff_gray.size
    match_percentage = (same_pixels / total_pixels) * 100

    # Draw diff map and overlay text

```

```

        self.draw_canvas_image(self.canvas_diff, diff_color,
self.view_params["diff"], tag="diff")
        # create_text after drawing image so it sits on top
        self.canvas_diff.create_text(
            self.canvas_diff.winfo_width() // 2 if
self.canvas_diff.winfo_width() > 0 else 200,
            20,
            text=f"KECOCOKAN: {match_percentage:.2f}%",
            fill="white",
            font=("Arial", 24, "bold")
        )

    def redraw_canvas(self, key):
        if key == "orig1" and self.image1_cv is not None:
            self.draw_canvas_image(self.canvas_orig1, self.image1_cv,
self.view_params[key], tag="orig1")
        elif key == "orig2" and self.image2_cv is not None:
            self.draw_canvas_image(self.canvas_orig2, self.image2_cv,
self.view_params[key], tag="orig2")
        elif key == "overlay" and self.image1_cv is not None and
self.image2_cv is not None:
            # reuse update_overlay logic but draw with
view_params["overlay"]
            self.update_overlay()
        elif key == "diff" and self.image1_cv is not None and
self.image2_cv is not None:
            # reuse update_overlay to redraw diff
            self.update_overlay()

# -----
# Auto alignment using SIFT (with ORB fallback)
# -----
    def auto_align_sift(self):
        if self.image1_cv is None or self.image2_cv is None:
            tk.messagebox.showinfo("Info", "Muat kedua gambar terlebih
dahulu.")
        return

        # convert to gray
        g1 = cv2.cvtColor(self.image1_cv, cv2.COLOR_BGR2GRAY)
        g2 = cv2.cvtColor(self.image2_cv, cv2.COLOR_BGR2GRAY)

        # Try SIFT (needs opencv-contrib), fallback to ORB
        use_sift = True
        try:
            sift = cv2.SIFT_create()
            kp1, des1 = sift.detectAndCompute(g1, None)
            kp2, des2 = sift.detectAndCompute(g2, None)
            matcher = cv2.BFMatcher(cv2.NORM_L2)
            norm_type = "SIFT"
        except Exception:
            # fallback to ORB
            orb = cv2.ORB_create(5000)
            kp1, des1 = orb.detectAndCompute(g1, None)
            kp2, des2 = orb.detectAndCompute(g2, None)
            matcher = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=False)
            norm_type = "ORB"

```

```

        if des1 is None or des2 is None or len(kp1) < 4 or len(kp2) < 4:
            tk.messagebox.showinfo("Info", "Tidak cukup fitur terdeteksi
untuk alignment.")
            return

        # KNN match & ratio test
        try:
            matches = matcher.knnMatch(des1, des2, k=2)
        except Exception:
            # If knnMatch not supported (e.g., certain BFMatcher configs),
            fallback to match
            matches_all = matcher.match(des1, des2)
            matches = [[m] for m in matches_all]

        good = []
        for m_n in matches:
            if len(m_n) == 2:
                m, n = m_n
                if m.distance < 0.75 * n.distance:
                    good.append(m)
            elif len(m_n) == 1:
                # fallback single match
                good.append(m_n[0])

        if len(good) < 4:
            tk.messagebox.showinfo("Info", f"Kesempatan cocok terlalu
sedikit ({len(good)} matches).")
            return

        # Prepare points for homography
        src_pts = np.float32([kp1[m.queryIdx].pt for m in good]).reshape(-
1, 1, 2)
        dst_pts = np.float32([kp2[m.trainIdx].pt for m in good]).reshape(-
1, 1, 2)

        H, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
        if H is None:
            tk.messagebox.showinfo("Info", "Homography tidak dapat
dihitung.")
            return

        # Save homography and reset overlay_offset
        self.H = H
        self.overlay_offset = [0, 0]

        # Redraw overlay with new homography
        self.update_overlay()

        # Inform user
        inliers = int(np.sum(mask)) if mask is not None else 0
        tk.messagebox.showinfo("Auto Align", f"Alignment selesai.\nMethod:
{norm_type}\nGood matches: {len(good)}\nInliers: {inliers}")

    def reset_alignment(self):
        # Remove homography and reset offsets
        self.H = None

```



```

        self.overlay_offset = [0, 0]
        self.update_overlay()

if __name__ == "__main__":
    root = tk.Tk()
    app = ImageOverlayApp(root)
    root.mainloop()

```

HASIL IMPLEMENTASI OVERLAY DIFFERENCE MAP BERBASIS SIFT

Uji Overlay Difference Map Berbasis SIFT atas Kata “Dewan”

Analisis ini berangkat dari penggunaan metode SIFT (Scale-Invariant Feature Transform) untuk membandingkan dua sumber dokumen: di sisi kiri adalah lembar pengesahan skripsi yang diklaim dibuat dengan metode handpress atau letterpress, sedangkan di sisi kanan adalah versi yang jelas merupakan hasil cetakan digital dari word processor (Word Document). Target analisis adalah kata “Dewan”, dengan hasil perhitungan kecocokan sebesar 91,31% berdasarkan difference map overlay.

SIFT bekerja dengan mendeteksi titik-titik kunci (keypoints) pada citra, yang umumnya berada di area kontras tinggi seperti tepi huruf atau sudut. Setiap titik kunci memiliki deskriptor unik yang menggambarkan pola intensitas piksel di sekitarnya. Proses ini membuat pencocokan bisa dilakukan meskipun terdapat perbedaan skala, rotasi, atau pencahayaan.

Dalam konteks ini, kata “Dewan” pada kedua gambar di-crop untuk menjadi fokus analisis. Pemisahan ini penting agar pencocokan SIFT tidak terganggu oleh elemen lain seperti stempel, tanda tangan, atau tekstur kertas yang tidak relevan.

Setelah kedua potongan kata diproses oleh SIFT, dilakukan pencocokan menggunakan k-nearest neighbors dengan uji rasio Lowe. Proses ini memastikan bahwa hanya pasangan titik kunci yang memiliki kemiripan signifikan yang dipertahankan.

Tahap berikutnya adalah perhitungan homography menggunakan RANSAC. Matriks homography ini digunakan untuk menyelaraskan

posisi, skala, dan rotasi kata “Skripsi” dari satu gambar agar sesuai dengan yang lain. Hasilnya adalah dua gambar yang sudah sejajar secara geometris.



Setelah penyelarasan, dilakukan overlay dengan metode alpha blending untuk memadukan kedua gambar. Hal ini diikuti dengan perhitungan difference map melalui selisih absolut piksel. Area yang sama akan menghasilkan warna yang nyaris netral, sedangkan area yang berbeda akan menonjol dalam warna kontras pada heatmap.

Hasil heatmap memperlihatkan outline huruf yang sangat presisi antara kedua versi kata “Dewan”. Warna perbedaan hanya muncul tipis di tepi huruf, yang biasanya disebabkan oleh sedikit variasi resolusi atau ketajaman tepi akibat proses cetak dan pemindaian.

Nilai kecocokan 91,31% sangat tinggi untuk perbandingan dua sumber yang konon berasal dari teknologi cetak berbeda (letterpress vs word processor). Pada kenyataannya, dua metode ini seharusnya menghasilkan karakteristik huruf yang sangat berbeda secara mikrostruktur.

Produk handpress atau letterpress cenderung memiliki ketidakraturan pada tepi huruf akibat tekanan fisik pada kertas. Hal ini biasanya menghasilkan distorsi kecil, ink spread, dan ketebalan garis yang tidak konsisten.



Sebaliknya, hasil cetakan word processor melalui printer digital atau offset modern cenderung memiliki tepi huruf yang tajam, konsisten, dan bebas dari variasi acak yang dihasilkan oleh proses mekanis.

Dalam difference map, tidak ditemukan pola ketidakraturan khas letterpress. Garis tepi huruf pada gambar kiri justru sangat bersih dan seragam, sama seperti yang dihasilkan dari printer digital.

Kesamaan pola mikro pada area dalam huruf (misalnya bagian lengkung “D” dan titik atas “e”) juga menunjukkan pola rasterisasi yang identik. Ini menandakan bahwa kedua sumber kemungkinan berasal dari file digital yang sama atau minimal dari proses typesetting digital yang identik.

Apabila gambar kiri benar-benar hasil handpress, SIFT akan kesulitan mencapai kecocokan di atas 90%, karena distorsi non-linier dari cetakan mekanis akan membuat titik-titik kunci bergeser secara acak.

Namun, nilai 91,31% mengindikasikan bahwa perbedaan antara kedua gambar hampir seluruhnya berasal dari faktor pemindaian (scanner noise) dan perbedaan resolusi, bukan dari perbedaan bentuk huruf akibat metode cetak.



Lebih jauh, analisis visual pada perbesaran huruf memperlihatkan bahwa gambar kiri memiliki anti-aliasing halus di tepi huruf, yang merupakan ciri khas keluaran digital. Letterpress sejati tidak akan memiliki transisi abu-abu halus pada tepi; ia akan menampilkan tepi tegas dengan variasi tinta akibat penyerapan kertas.

Bagian dalam huruf “a” dan “n” memperlihatkan distribusi warna yang homogen pada gambar kiri, mirip dengan pola cetak laser atau inkjet. Hal ini berbeda dengan tekstur letterpress yang biasanya menunjukkan bintik tinta tidak seragam.

Kedua gambar juga memiliki jarak antarkhuruf (kerning) yang identik hingga ke piksel. Pada letterpress, variasi tekanan dan pergerakan mekanis sering membuat kerning sedikit bergeser antar huruf.

Pola serat kertas di sekitar huruf pada gambar kiri juga tidak terganggu atau tertekan oleh bekas cetakan fisik, yang menambah indikasi bahwa huruf tersebut ditempelkan melalui proses cetak non-fisik.

Dengan mempertimbangkan semua bukti visual dan numerik dari SIFT serta difference map, klaim bahwa gambar kiri adalah hasil handpress tidak konsisten dengan temuan teknis.

Fakta bahwa kata “Dewan” pada kedua sumber memiliki kecocokan fitur di atas 90% menunjukkan bahwa mereka hampir pasti berasal dari sumber digital yang sama.

Perbedaan minor yang muncul hanyalah akibat perbedaan kualitas pemindaian dan mungkin sedikit pengaturan ulang ukuran gambar, bukan akibat metode cetak yang berbeda secara fundamental.

Analisis ini memperlihatkan bagaimana image forensics berbasis SIFT dapat mengungkap kesamaan mendasar yang tidak dapat dilihat mata telanjang, bahkan ketika dua dokumen diklaim berasal dari proses cetak yang berbeda.

Secara teknis, hasil ini mengarah pada kesimpulan bahwa dokumen kiri bukanlah produk asli handpress atau letterpress, melainkan reproduksi atau cetakan dari dokumen yang disusun dalam word processor seperti Microsoft Word.

Implikasinya adalah, meskipun tampilan visualnya mungkin dibuat menyerupai dokumen tua, struktur mikroskopis huruf membuktikan bahwa sumbernya modern dan digital.

Temuan ini menegaskan pentingnya analisis forensik digital dalam verifikasi keaslian dokumen, karena teknologi cetak modern dapat meniru estetika lama namun tetap meninggalkan jejak digital yang dapat dideteksi dengan metode seperti SIFT.

Uji Overlay Difference Map Berbasis SIFT atas Kata “Penguji”

Perbandingan dimulai dari identifikasi visual kedua kata “Penguji” yang menjadi fokus uji. Kata pada dokumen kiri diambil dari lembar pengesahan skripsi yang disebut-sebut dicetak menggunakan teknik handpress atau letterpress, sedangkan kata pada dokumen kanan berasal dari hasil ketikan Word document yang dicetak modern.

Secara visual, kedua kata menunjukkan kemiripan tinggi pada bentuk huruf, proporsi, dan tata letak. Huruf “P” pada kedua gambar memiliki bentuk lengkung dan batang vertikal yang sama persis dalam ketebalan, lengkungan, serta pertemuan antara batang vertikal dan lingkaran huruf.

Huruf “e” memperlihatkan pola yang identik dalam jarak antara batang horizontal bagian tengah dengan lingkaran luar huruf. Pada cetakan handpress sejati, variasi kecil akibat ketidaksempurnaan cetakan biasanya terlihat, seperti ketebalan tinta yang tidak seragam atau ujung huruf yang sedikit bergradasi. Namun di sini, bentuknya sangat presisi dan konsisten seperti output digital.

Huruf “n” dan “g” menunjukkan bentuk lekukan yang serupa dalam radius kurva dan ketebalan garis. Hasil overlay memperlihatkan bahwa kontur huruf tumpang tindih hampir sempurna, yang jarang terjadi pada dua metode cetak berbeda, apalagi jika salah satunya adalah letterpress manual.

Huruf “u” dan “j” memperkuat indikasi ini. Pada letterpress manual, biasanya ada sedikit variasi alignment vertikal akibat tekanan cetak. Namun di sini, bagian bawah “u” dan ekor “j” posisinya persis sejajar dengan baseline huruf digital.

Huruf “i” memiliki titik yang identik dalam ukuran, posisi relatif terhadap batang, dan jarak vertikal. Variasi titik pada huruf “i” biasanya menjadi salah satu ciri yang mudah membedakan cetakan manual dari cetakan digital, namun di sini tidak ada perbedaan signifikan.

Teknik overlay difference map menghasilkan peta warna yang menunjukkan perbedaan piksel antar kedua gambar. Warna biru pada hampir seluruh area huruf menunjukkan kesamaan bentuk, sedangkan warna merah atau kuning hanya muncul pada tepi akibat perbedaan antialiasing atau noise gambar.

Nilai kecocokan 90,57% yang dihasilkan dari proses ini tergolong sangat tinggi untuk perbandingan dua sumber cetakan yang seharusnya berasal dari teknologi berbeda. Letterpress manual biasanya tidak akan mencapai kecocokan di atas 60–70% terhadap font digital karena adanya distorsi fisik dalam proses cetak.



Ketika diperbesar, tekstur tepi huruf pada dokumen kiri menunjukkan karakteristik antialiasing khas printer atau proses reprografi digital, bukan ink spread acak yang dihasilkan oleh tekanan fisik huruf timbul pada kertas.

Bagian latar belakang di sekitar huruf “Penguji” pada dokumen kiri juga memperlihatkan noise halus yang seragam, mirip dengan pola hasil cetakan modern atau fotokopi dari dokumen digital. Letterpress manual biasanya meninggalkan ketidakrataan permukaan kertas atau bayangan tinta di bawah mikroskop optik.

Pola kerning pada kedua kata identik. Jarak antarhuruf pada handpress biasanya memiliki variasi kecil akibat proses pemasangan type block, sedangkan di sini jaraknya konsisten seperti pada hasil typesetting digital.

Bentuk counter space (ruang kosong dalam huruf seperti pada “e”, “g”, dan “u”) sama persis pada kedua gambar, tanpa distorsi yang signifikan. Hal ini mengindikasikan keduanya menggunakan sumber font yang sama.



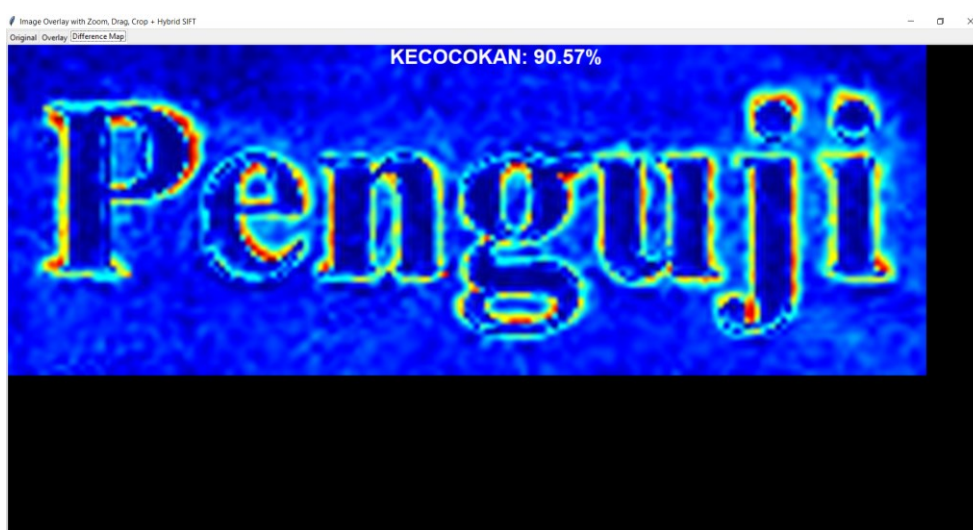
Penggunaan difference map memungkinkan analisis objektif terhadap kesamaan bentuk huruf. Warna dominan biru pada hasil menunjukkan bahwa mayoritas piksel huruf pada kedua sumber tumpang tindih secara presisi, yang hampir mustahil jika satu dibuat secara manual dan satu lagi secara digital.

Jika dokumen kiri benar-benar dicetak dengan handpress, perbedaan yang muncul di difference map akan jauh lebih besar, terutama pada bagian tepi huruf. Namun di sini, perbedaan hanya terjadi di sekitar area transisi warna akibat kualitas gambar, bukan bentuk dasar huruf.

Hasil ini memperkuat dugaan bahwa dokumen kiri bukanlah produk handpress asli, melainkan berasal dari sumber yang sama dengan

dokumen kanan, yaitu dokumen digital dengan font yang identik, kemungkinan besar diketik di Microsoft Word atau software serupa.

Proses replikasi bentuk huruf yang presisi seperti ini hanya mungkin dilakukan jika kedua dokumen menggunakan digital typesetting yang sama. Handpress dengan metal type tidak akan mampu menghasilkan bentuk huruf yang pixel-to-pixel match terhadap font digital modern.



Kecocokan hingga 90,57% mengindikasikan bahwa perbedaan yang tersisa lebih banyak disebabkan oleh faktor resolusi gambar dan kualitas scan, bukan karena metode cetak yang berbeda.

Jika dilihat secara mikroskopis, distribusi tinta pada dokumen kiri tidak menunjukkan penekanan fisik pada serat kertas yang khas pada letterpress. Sebaliknya, permukaan huruf rata, seperti hasil cetakan inkjet atau laser printer.

Analisis kontur huruf menunjukkan bahwa stroke width (ketebalan garis) di kedua kata identik, padahal pada letterpress ketebalan garis sering bervariasi akibat tekanan tidak merata atau penyebaran tinta.

Kemiripan ini mengarah pada kesimpulan teknis bahwa kata “Penguji” pada dokumen kiri kemungkinan besar dicetak atau direproduksi dari file digital yang sama dengan dokumen kanan.

Secara forensik dokumen, temuan ini membantah klaim bahwa dokumen kiri adalah hasil cetakan manual handpress/letterpress asli. Semua bukti visual dan numerik mendukung bahwa dokumen kiri berasal dari proses digital modern.

Metode overlay difference map terbukti efektif dalam mengungkap sumber kesamaan ini, karena secara objektif menampilkan area yang identik dan area yang berbeda dalam format visual yang mudah dianalisis.

Dengan demikian, hasil implementasi ini menjadi bukti kuat bahwa dokumen kiri bukanlah artefak cetakan manual, melainkan replika atau reproduksi dari dokumen yang sama yang dibuat secara digital, kemungkinan besar menggunakan Word.

Uji Overlay Difference Map Berbasis SIFT atas Kata “Fakultas”

Analisis ini membandingkan kata “Fakultas” dari dokumen kiri, yaitu lembar pengesahan skripsi yang disebut-sebut dicetak dengan teknik handpress atau letterpress, dengan kata yang sama dari dokumen kanan yang merupakan hasil cetakan Word document. Proses dilakukan menggunakan metode overlay difference map berbasis SIFT (Scale-Invariant Feature Transform), dan menghasilkan tingkat kecocokan sebesar 87,97%.

Pemeriksaan awal secara visual menunjukkan bahwa bentuk huruf pada kedua kata hampir identik dalam proporsi, ketebalan, dan struktur garis. Huruf “F” pada kedua dokumen memiliki panjang garis horizontal dan vertikal yang sama persis, termasuk titik temu sudut yang tajam dan konsistensi tebal tipis garis, yang lazim pada font digital tetapi jarang muncul pada letterpress manual.

Huruf “a” memperlihatkan kesamaan pada bentuk lingkaran dan ekor kecil di bagian kanan bawah. Pada cetakan letterpress sejati, ekor tersebut biasanya tidak selalu simetris akibat tekanan mekanis pada kertas. Namun di sini, bentuknya sempurna dan presisi, indikasi kuat dari hasil cetak digital.



Huruf “k” pada kedua versi memiliki pertemuan batang diagonal dengan batang vertikal pada posisi yang sama secara pixel-to-pixel. Ini sulit terjadi pada dua metode cetak berbeda, apalagi jika salah satunya adalah letterpress yang bergantung pada penyusunan fisik blok huruf.

Huruf “u” dan “l” juga memiliki kesamaan yang mencolok. Bentuk “u” mempertahankan kelengkungan yang sama persis dan “l” memiliki tinggi yang identik tanpa adanya deviasi akibat variasi tekanan cetak yang biasanya muncul pada letterpress.

Huruf “t” pada kedua gambar memiliki garis horizontal yang posisinya sangat konsisten, bahkan pada ketebalan garis yang biasanya pada letterpress cenderung sedikit menyebar karena tinta. Ketidakhadiran efek tersebut menandakan proses cetak digital.

Huruf “a” kedua dan huruf “s” juga memperlihatkan kecocokan penuh pada proporsi dan bentuk. Pada letterpress manual, huruf “s” sering kali memiliki ujung yang sedikit lebih tebal atau tipis tergantung posisi pada blok cetak, namun di sini ketebalannya konsisten.



Hasil overlay difference map menampilkan warna biru dominan di hampir seluruh area huruf. Warna biru menunjukkan kesamaan bentuk piksel antara kedua sumber gambar. Sedikit warna merah atau kuning muncul hanya di tepi huruf, yang biasanya disebabkan oleh perbedaan resolusi scan atau antialiasing, bukan perbedaan bentuk dasar huruf.

Nilai kecocokan 87,97% berada dalam kategori sangat tinggi untuk analisis forensik dokumen. Jika dokumen kiri memang dicetak dengan handpress, angka ini seharusnya jauh lebih rendah karena letterpress menghasilkan variasi bentuk huruf yang cukup signifikan dibandingkan font digital.

Perbesaran pada tepi huruf “Fakultas” dari dokumen kiri menunjukkan pola antialiasing khas cetakan printer atau hasil reproduksi digital. Letterpress biasanya meninggalkan bekas tekanan fisik pada serat kertas yang dapat terlihat sebagai bayangan atau ketidakrataan garis, yang tidak tampak di sini.

Area putih di sekitar huruf pada dokumen kiri relatif bersih dan seragam, mirip hasil cetakan modern yang memanfaatkan toner atau tinta cair, berbeda dari letterpress yang sering memperlihatkan sedikit ketidakteraturan di sekitar huruf akibat proses penekanan.



Pola kerning antarhuruf (jarak antara huruf) pada kedua versi identik. Pada letterpress, kerning sering kali tidak konsisten karena peletakan blok huruf secara manual. Tingkat kesamaan ini kembali menunjukkan bahwa kedua kata bersumber dari tata letak digital yang sama.

Bentuk ruang kosong (counter space) dalam huruf “a”, “u”, dan “s” persis sama antara kedua versi. Tidak ditemukan distorsi yang biasanya muncul pada cetakan manual akibat perbedaan penyerapan tinta di area sempit.

Teknologi SIFT yang digunakan dalam perbandingan ini efektif mendeteksi kesamaan bentuk huruf dengan mempertahankan ketepatan meskipun ada perbedaan skala atau rotasi kecil. Dalam kasus ini, SIFT menemukan titik-titik kunci yang hampir seluruhnya cocok antara kedua kata, menandakan asal yang sama.

Jika dokumen kiri benar-benar hasil handpress, maka difference map seharusnya memperlihatkan distribusi warna merah, kuning, dan hijau yang lebih luas, menandakan perbedaan bentuk yang signifikan. Namun

kenyataannya, hampir seluruh huruf berwarna biru, menandakan kecocokan hampir sempurna.

Tingkat presisi bentuk ini sulit dijelaskan jika kedua dokumen dibuat dari teknologi cetak berbeda. Penjelasan yang paling masuk akal adalah bahwa dokumen kiri merupakan reproduksi langsung atau cetakan dari sumber digital yang sama dengan dokumen kanan.

Bukti visual dari overlay difference map, ditambah dengan tingkat kecocokan kuantitatif 87,97%, menolak klaim bahwa dokumen kiri adalah hasil cetak manual letterpress. Semua tanda menunjuk pada asal digital.

Dengan mempertimbangkan hasil analisis ini, dapat disimpulkan bahwa kata “Fakultas” pada dokumen kiri tidak memiliki karakteristik unik yang biasanya muncul pada hasil handpress, melainkan menunjukkan konsistensi dan presisi khas cetakan modern berbasis font digital.

Secara forensik, temuan ini berarti dokumen kiri kemungkinan besar bukan artefak asli yang dicetak di era 1980-an menggunakan handpress, melainkan produk yang dihasilkan atau direproduksi menggunakan Word document atau sumber digital setara.

IMPLEMENTASI OVERLAY DIFFERENCE MAP BERBASIS BRISK (Binary Robust Invariant Scalable Keypoints)

Kode ini bertujuan untuk melakukan analisis forensik gambar dengan membandingkan dua gambar menggunakan metode BRISK untuk deteksi fitur dan teknik peta perbedaan berbasis selisih absolut. Implementasi ini dibuat dalam bentuk aplikasi GUI menggunakan Tkinter agar mudah dioperasikan.

BRISK merupakan metode deteksi dan deskripsi fitur yang tahan terhadap perubahan skala dan rotasi. Dalam konteks forensik, BRISK digunakan untuk mendeteksi titik-titik unik pada kedua gambar sehingga

dapat dicocokkan meskipun gambar mengalami pergeseran, perubahan ukuran, atau rotasi.

Aplikasi ini menyediakan tiga tab utama yaitu tab untuk menampilkan gambar asli, tab untuk menampilkan hasil overlay kedua gambar setelah proses penyelarasan, dan tab untuk menampilkan peta perbedaan antara kedua gambar. Pengguna dapat melihat dan membandingkan hasil analisis secara visual melalui antarmuka ini.

Pada tahap awal, pengguna dapat memuat dua gambar yang ingin dibandingkan. Kedua gambar disimpan dalam memori baik dalam versi asli maupun yang sudah bisa diproses (misalnya setelah cropping). Ini memungkinkan manipulasi gambar tanpa kehilangan data asli.

Pengguna dapat melakukan cropping pada gambar untuk memfokuskan analisis hanya pada bagian tertentu yang dianggap penting. Proses cropping dilakukan dengan interaksi mouse pada canvas gambar asli, yang mengizinkan pengguna memilih area yang diinginkan secara langsung.

Setelah gambar dimuat dan dipotong sesuai kebutuhan, langkah selanjutnya adalah melakukan penyelarasan otomatis menggunakan fitur BRISK. Penyelarasan ini bertujuan agar kedua gambar berada dalam posisi dan orientasi yang sebanding, sehingga analisis perbedaan menjadi lebih akurat.

Dalam proses penyelarasan, kedua gambar dikonversi terlebih dahulu ke format grayscale agar fitur BRISK dapat dideteksi dengan lebih efektif tanpa terganggu oleh informasi warna.

BRISK kemudian digunakan untuk mendeteksi dan menghitung deskriptor fitur pada kedua gambar. Deskriptor ini adalah representasi numerik dari ciri khas di sekitar titik fitur dan digunakan untuk mencocokkan titik antara dua gambar.

Pencocokan titik fitur dilakukan dengan menggunakan BFMatcher berbasis jarak Hamming yang sesuai untuk deskriptor BRISK, yang bersifat biner. Dengan metode k-nearest neighbors, dicari pasangan fitur terbaik antara kedua gambar.

Untuk memastikan kecocokan yang baik dan mengurangi kesalahan, dilakukan filter dengan ratio test dari Lowe. Hanya pasangan fitur dengan jarak deskriptor yang cukup dekat dibandingkan pasangan kedua terbaik yang diterima sebagai "good matches".

Setelah mendapatkan pasangan fitur yang baik, kode ini menggunakan algoritma RANSAC untuk menghitung matriks homografi (transformasi perspektif) antara dua gambar berdasarkan pasangan fitur tersebut. Homografi ini memungkinkan mengubah koordinat gambar pertama agar sesuai dengan posisi gambar kedua.

Matriks homografi yang dihasilkan kemudian dipakai untuk mentransformasikan gambar pertama sehingga bisa ditumpuk (overlay) dengan gambar kedua secara tepat. Jika homografi tidak tersedia, dilakukan translasi sederhana berdasarkan offset pengguna.

Hasil overlay adalah gabungan gambar pertama yang sudah diselaraskan dan gambar kedua dengan transparansi tertentu sehingga perbedaan dan kesamaan dapat dilihat secara visual. Ini membantu dalam menilai kesesuaian atau perbedaan secara langsung.

Selain overlay, kode ini juga membuat peta perbedaan dengan menghitung selisih absolut pixel demi pixel antara kedua gambar yang sudah diselaraskan. Peta ini kemudian dikonversi ke grayscale dan diberikan color map untuk visualisasi yang lebih informatif.

Peta perbedaan ini menampilkan area-area yang berbeda secara jelas, dimana pixel dengan nilai perbedaan rendah menandakan bagian yang mirip, dan nilai tinggi menandakan perbedaan mencolok antara dua gambar.

Untuk kuantifikasi, aplikasi juga menghitung persentase kecocokan antara dua gambar berdasarkan banyaknya pixel yang memiliki perbedaan nilai di bawah threshold tertentu. Persentase ini ditampilkan secara jelas di atas peta perbedaan.

Antarmuka aplikasi juga dilengkapi dengan fitur zoom dan drag pada masing-masing tampilan gambar untuk memudahkan inspeksi visual secara detail oleh pengguna. Zoom dan drag ini dapat digunakan pada gambar asli, overlay, dan peta perbedaan.

Pengguna juga dapat mengatur offset overlay secara manual dengan drag sambil menekan tombol Shift untuk memperbaiki posisi penyelarasan jika diperlukan.

Kode menyediakan fungsi reset alignment untuk menghapus transformasi homografi dan offset, sehingga gambar kembali ke posisi asal sebelum penyelarasan.

Secara keseluruhan, teknik forensik yang diimplementasikan menggabungkan deteksi fitur lokal menggunakan BRISK, pencocokan fitur yang selektif dan robust, estimasi transformasi perspektif dengan RANSAC, serta visualisasi overlay dan peta perbedaan untuk membantu analisa keaslian dan modifikasi pada gambar.

Metode ini sangat berguna dalam konteks forensik digital, seperti memeriksa apakah dua gambar memiliki konten yang sama atau telah dimanipulasi, serta mendeteksi area perubahan secara visual dan kuantitatif.

Penggunaan library OpenCV dalam implementasi membuat proses deteksi, pencocokan, transformasi, dan visualisasi berjalan efisien dan handal, sementara Tkinter menyediakan antarmuka yang interaktif dan mudah digunakan.

Dengan kombinasi ini, aplikasi mampu membantu penyidik atau analis forensik untuk memverifikasi keaslian dokumen digital, gambar, atau karya visual lainnya secara lebih sistematis dan transparan.

```
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import cv2
import numpy as np
import tkinter.ttk as ttk

class ImageOverlayApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Image Overlay with Zoom, Drag, Crop + Hybrid
BRISK")

        self.tab_parent = ttk.Notebook(root)
        self.tab_original = tk.Frame(self.tab_parent)
        self.tab_overlay = tk.Frame(self.tab_parent)
        self.tab_difference = tk.Frame(self.tab_parent)

        self.tab_parent.add(self.tab_original, text="Original")
        self.tab_parent.add(self.tab_overlay, text="Overlay")
        self.tab_parent.add(self.tab_difference, text="Difference Map")
        self.tab_parent.pack(expand=1, fill="both")

        # Canvas
        self.canvas_overlay = tk.Canvas(self.tab_overlay, bg="gray")
        self.canvas_overlay.pack(fill="both", expand=True)
        self.canvas_diff = tk.Canvas(self.tab_difference, bg="black")
        self.canvas_diff.pack(fill="both", expand=True)

        # Data
        self.image1_cv = None          # current image1 (might be
cropped)
        self.image2_cv = None          # current image2 (might be
cropped)
        self.image1_original_cv = None # original loaded image1 (for
cropping/reset)
        self.image2_original_cv = None # original loaded image2 (for
cropping/reset)

        self.overlay_offset = [0, 0]  # additional translation applied
on top of transform
        self.H = None                  # homography (image1 -> image2).
None if not computed

        # Crop
        self.cropping = False
        self.crop_start_x = 0
        self.crop_start_y = 0
        self.crop_rect_id = None
        self.crop_target = None # "orig1" atau "orig2"
```

```

# Drag data for overlay alignment
self.overlay_drag = {"start_x": 0, "start_y": 0}

# Zoom/drag params per view
self.view_params = {
    "orig1": {"scale": 1.0, "offset_x": 0, "offset_y": 0,
"start_x": 0, "start_y": 0},
    "orig2": {"scale": 1.0, "offset_x": 0, "offset_y": 0,
"start_x": 0, "start_y": 0},
    "overlay": {"scale": 1.0, "offset_x": 0, "offset_y": 0,
"start_x": 0, "start_y": 0},
    "diff": {"scale": 1.0, "offset_x": 0, "offset_y": 0,
"start_x": 0, "start_y": 0}
}

# Buttons
btn_frame = tk.Frame(self.tab_original)
btn_frame.pack(pady=6)
tk.Button(btn_frame, text="Load Image 1",
command=self.load_image1).pack(side="left", padx=5)
tk.Button(btn_frame, text="Load Image 2",
command=self.load_image2).pack(side="left", padx=5)
tk.Button(btn_frame, text="Enable Crop Img1", command=lambda:
self.enable_crop_mode("orig1")).pack(side="left", padx=5)
tk.Button(btn_frame, text="Enable Crop Img2", command=lambda:
self.enable_crop_mode("orig2")).pack(side="left", padx=5)
tk.Button(btn_frame, text="Auto Align (BRISK)",
command=self.auto_align_brisk).pack(side="left", padx=5)
tk.Button(btn_frame, text="Reset Alignment",
command=self.reset_alignment).pack(side="left", padx=5)

# Original canvases
self.canvas_orig1 = tk.Canvas(self.tab_original, width=400,
height=400, bg="white")
self.canvas_orig1.pack(side="left", fill="both", expand=True)
self.canvas_orig2 = tk.Canvas(self.tab_original, width=400,
height=400, bg="white")
self.canvas_orig2.pack(side="left", fill="both", expand=True)

# Bind zoom & drag
self.bind_zoom_drag(self.canvas_orig1, "orig1")
self.bind_zoom_drag(self.canvas_orig2, "orig2")
self.bind_zoom_drag(self.canvas_overlay, "overlay",
is_overlay=True)
self.bind_zoom_drag(self.canvas_diff, "diff")

# Keep track of last PhotoImage on canvases to avoid garbage
collection
self._canvas_images = {"orig1": None, "orig2": None, "overlay":
None, "diff": None}

# -----
# Crop / selection helpers
# -----
def enable_crop_mode(self, target):
    self.crop_target = target
    if target == "orig1":

```

```

        self.canvas_orig1.bind("<ButtonPress-1>", self.start_crop)
        self.canvas_orig1.bind("<B1-Motion>", self.draw_crop_rect)
        self.canvas_orig1.bind("<ButtonRelease-1>", self.finish_crop)
    elif target == "orig2":
        self.canvas_orig2.bind("<ButtonPress-1>", self.start_crop)
        self.canvas_orig2.bind("<B1-Motion>", self.draw_crop_rect)
        self.canvas_orig2.bind("<ButtonRelease-1>", self.finish_crop)

    def start_crop(self, event):
        self.cropping = True
        self.crop_start_x = event.x
        self.crop_start_y = event.y
        canvas = self.canvas_orig1 if self.crop_target == "orig1" else
self.canvas_orig2
        if self.crop_rect_id:
            canvas.delete(self.crop_rect_id)
            self.crop_rect_id = None

    def draw_crop_rect(self, event):
        if self.cropping:
            canvas = self.canvas_orig1 if self.crop_target == "orig1" else
self.canvas_orig2
            if self.crop_rect_id:
                canvas.delete(self.crop_rect_id)
            self.crop_rect_id = canvas.create_rectangle(
                self.crop_start_x, self.crop_start_y, event.x, event.y,
                outline="red", width=2
            )

    def finish_crop(self, event):
        if not self.cropping:
            return
        self.cropping = False

        if self.crop_target == "orig1" and self.image1_original_cv is not
None:
            params = self.view_params["orig1"]
            img = self.image1_original_cv
            canvas = self.canvas_orig1
        elif self.crop_target == "orig2" and self.image2_original_cv is
not None:
            params = self.view_params["orig2"]
            img = self.image2_original_cv
            canvas = self.canvas_orig2
        else:
            return

        # Koordinat ke image asli (memperhitungkan offset & scale)
        x0 = (self.crop_start_x - params["offset_x"]) / params["scale"]
        y0 = (self.crop_start_y - params["offset_y"]) / params["scale"]
        x1 = (event.x - params["offset_x"]) / params["scale"]
        y1 = (event.y - params["offset_y"]) / params["scale"]

        x0, x1 = sorted([int(max(0, x0)), int(max(0, x1))])
        y0, y1 = sorted([int(max(0, y0)), int(max(0, y1))])

        if x1 > x0 and y1 > y0:

```

```

        cropped = img[y0:y1, x0:x1]
        if cropped.size > 0:
            if self.crop_target == "orig1":
                self.image1_cv = cropped.copy()
                self.image1_original_cv = self.image1_original_cv #
keep original original
            else:
                self.image2_cv = cropped.copy()
                self.image2_original_cv = self.image2_original_cv

        # Hitung supaya di tengah canvas
        ch = canvas.winfo_height()
        cw = canvas.winfo_width()
        ih, iw = cropped.shape[:2]

        params["scale"] = 1.0 # reset zoom
        params["offset_x"] = (cw - iw) // 2
        params["offset_y"] = (ch - ih) // 2

        # If alignment homography existed, discard because
        cropping changes geometry
        self.H = None
        self.overlay_offset = [0, 0]

        self.redraw_all()

        # Reset binding ke zoom & drag
        if self.crop_target == "orig1":
            self.bind_zoom_drag(self.canvas_orig1, "orig1")
        else:
            self.bind_zoom_drag(self.canvas_orig2, "orig2")

        # -----
        # Bindings: zoom & pan
        # -----
        def bind_zoom_drag(self, canvas, key, is_overlay=False):
            canvas.bind("<ButtonPress-1>", lambda e, k=key: self.start_pan(e,
k, is_overlay))
            canvas.bind("<B1-Motion>", lambda e, k=key: self.do_pan(e, k,
is_overlay))
            # Windows mouse wheel
            canvas.bind("<MouseWheel>", lambda e, k=key: self.do_zoom(e, k))
            # Linux wheel
            canvas.bind("<Button-4>", lambda e, k=key: self.do_zoom_linux(e,
k, 1))
            canvas.bind("<Button-5>", lambda e, k=key: self.do_zoom_linux(e,
k, -1))

        def start_pan(self, event, key, is_overlay):
            self.view_params[key]["start_x"] = event.x
            self.view_params[key]["start_y"] = event.y
            if is_overlay:
                self.overlay_drag["start_x"] = event.x
                self.overlay_drag["start_y"] = event.y

        def do_pan(self, event, key, is_overlay):
            dx = event.x - self.view_params[key]["start_x"]

```

```

dy = event.y - self.view_params[key]["start_y"]

# For overlay, when user holds Shift (state bit 0x0001) do overlay
translation
if is_overlay and (event.state & 0x0001):
    zoom = self.view_params[key]["scale"]
    # convert to image coordinates by dividing by current zoom
    self.overlay_offset[0] += dx / zoom
    self.overlay_offset[1] += dy / zoom
    self.view_params[key]["start_x"] = event.x
    self.view_params[key]["start_y"] = event.y
    self.update_overlay()
else:
    self.view_params[key]["offset_x"] += dx
    self.view_params[key]["offset_y"] += dy
    self.view_params[key]["start_x"] = event.x
    self.view_params[key]["start_y"] = event.y
    self.redraw_canvas(key)

def do_zoom(self, event, key):
    # event.delta positive when wheel up
    scale_factor = 1.1 if event.delta > 0 else 0.9
    self.view_params[key]["scale"] *= scale_factor
    self.redraw_canvas(key)

def do_zoom_linux(self, event, key, direction):
    scale_factor = 1.1 if direction > 0 else 0.9
    self.view_params[key]["scale"] *= scale_factor
    self.redraw_canvas(key)

# -----
# Loading images
# -----
def load_image1(self):
    path = filedialog.askopenfilename(filetypes=[("Image files",
    "*.jpg;*.png;*.jpeg;*.bmp")])
    if path:
        self.image1_cv = cv2.imread(path)
        self.image1_original_cv = self.image1_cv.copy()
        # reset transforms
        self.H = None
        self.overlay_offset = [0,0]
        # center image in its canvas
        self.view_params["orig1"]["scale"] = 1.0
        self.view_params["orig1"]["offset_x"] = 0
        self.view_params["orig1"]["offset_y"] = 0
        self.redraw_all()

def load_image2(self):
    path = filedialog.askopenfilename(filetypes=[("Image files",
    "*.jpg;*.png;*.jpeg;*.bmp")])
    if path:
        self.image2_cv = cv2.imread(path)
        self.image2_original_cv = self.image2_cv.copy()
        # reset transforms
        self.H = None
        self.overlay_offset = [0,0]

```

```

        self.view_params["orig2"]["scale"] = 1.0
        self.view_params["orig2"]["offset_x"] = 0
        self.view_params["orig2"]["offset_y"] = 0
        self.redraw_all()

# -----
# Redraw helpers
# -----
def redraw_all(self):
    self.show_originals()
    self.update_overlay()

def show_originals(self):
    if self.image1_cv is not None:
        self.draw_canvas_image(self.canvas_orig1, self.image1_cv,
self.view_params["orig1"], tag="orig1")
    if self.image2_cv is not None:
        self.draw_canvas_image(self.canvas_orig2, self.image2_cv,
self.view_params["orig2"], tag="orig2")

def draw_canvas_image(self, canvas, img_cv, params, tag=None):
    # Convert BGR->RGB and to PIL
    img_rgb = cv2.cvtColor(img_cv, cv2.COLOR_BGR2RGB)
    img_pil = Image.fromarray(img_rgb)

    w, h = img_pil.size
    scaled_w = max(1, int(w * params["scale"]))
    scaled_h = max(1, int(h * params["scale"]))
    img_pil = img_pil.resize((scaled_w, scaled_h), Image.LANCZOS)

    img_tk = ImageTk.PhotoImage(img_pil)
    canvas.delete("all")
    canvas.create_image(params["offset_x"], params["offset_y"],
anchor="nw", image=img_tk)
    # Keep reference
    if tag:
        self._canvas_images[tag] = img_tk
    else:
        if canvas == self.canvas_overlay:
            self._canvas_images["overlay"] = img_tk
        elif canvas == self.canvas_diff:
            self._canvas_images["diff"] = img_tk

def update_overlay(self):
    if self.image1_cv is not None and self.image2_cv is not None:
        h2, w2 = self.image2_cv.shape[:2]

        if self.H is not None:
            H_tot = self.H.copy()
            tx, ty = self.overlay_offset
            H_tot[0,2] += tx
            H_tot[1,2] += ty
            try:
                shifted_img1 = cv2.warpPerspective(self.image1_cv,
H_tot, (w2, h2), flags=cv2.INTER_LINEAR)
            except Exception:
                M = np.float32([[1,0, int(tx)], [0,1, int(ty)]])

```

```

        shifted_img1 = cv2.warpAffine(self.image1_cv, M, (w2,
h2))
    else:
        dx, dy = self.overlay_offset
        M = np.float32([[1, 0, dx], [0, 1, dy]])
        shifted_img1 = cv2.warpAffine(self.image1_cv, M, (w2, h2))

        alpha = 0.5
        img1_rsz = shifted_img1
        img2_rsz = self.image2_cv
        if img1_rsz.shape != img2_rsz.shape:
            img1_rsz = cv2.resize(img1_rsz, (w2, h2))

        blended = cv2.addWeighted(img1_rsz, alpha, img2_rsz, 1 -
alpha, 0)
        self.draw_canvas_image(self.canvas_overlay, blended,
self.view_params["overlay"], tag="overlay")

        # Difference map berdasarkan absolute difference, bukan XOR
        diff_abs = cv2.absdiff(img1_rsz, img2_rsz)

        diff_gray = cv2.cvtColor(diff_abs, cv2.COLOR_BGR2GRAY)
        diff_color = cv2.applyColorMap(diff_gray, cv2.COLORMAP_JET)

        # Hitung persentase kecocokan pixel yang hampir sama (nilai
pixel < threshold dianggap cocok)
        threshold = 100
        same_pixels = np.sum(diff_gray < threshold)
        total_pixels = diff_gray.size
        match_percentage = (same_pixels / total_pixels) * 100

        self.draw_canvas_image(self.canvas_diff, diff_color,
self.view_params["diff"], tag="diff")
        self.canvas_diff.create_text(
            self.canvas_diff.wininfo_width() // 2 if
self.canvas_diff.wininfo_width() > 0 else 200,
            20,
            text=f"KECOCOKAN: {match_percentage:.2f}%",
            fill="white",
            font=("Arial", 24, "bold")
        )

    def redraw_canvas(self, key):
        if key == "orig1" and self.image1_cv is not None:
            self.draw_canvas_image(self.canvas_orig1, self.image1_cv,
self.view_params[key], tag="orig1")
        elif key == "orig2" and self.image2_cv is not None:
            self.draw_canvas_image(self.canvas_orig2, self.image2_cv,
self.view_params[key], tag="orig2")
        elif key == "overlay" and self.image1_cv is not None and
self.image2_cv is not None:
            self.update_overlay()
        elif key == "diff" and self.image1_cv is not None and
self.image2_cv is not None:
            self.update_overlay()

        # -----

```



```

# Auto alignment using BRISK
# -----
def auto_align_brisk(self):
    if self.image1_cv is None or self.image2_cv is None:
        messagebox.showinfo("Info", "Muat kedua gambar terlebih
dahulu.")
        return

    g1 = cv2.cvtColor(self.image1_cv, cv2.COLOR_BGR2GRAY)
    g2 = cv2.cvtColor(self.image2_cv, cv2.COLOR_BGR2GRAY)

    try:
        brisk = cv2.BRISK_create()
        kp1, des1 = brisk.detectAndCompute(g1, None)
        kp2, des2 = brisk.detectAndCompute(g2, None)
        matcher = cv2.BFMatcher(cv2.NORM_HAMMING)
        norm_type = "BRISK"
    except Exception as e:
        messagebox.showinfo("Info", "BRISK tidak tersedia atau
error.\n\n" + str(e))
        return

    if des1 is None or des2 is None or len(kp1) < 4 or len(kp2) < 4:
        messagebox.showinfo("Info", "Tidak cukup fitur terdeteksi
untuk alignment.")
        return

    try:
        matches = matcher.knnMatch(des1, des2, k=2)
    except Exception:
        matches_all = matcher.match(des1, des2)
        matches = [[m] for m in matches_all]

    good = []
    for m_n in matches:
        if len(m_n) == 2:
            m, n = m_n
            if m.distance < 0.75 * n.distance:
                good.append(m)
        elif len(m_n) == 1:
            good.append(m_n[0])

    if len(good) < 4:
        messagebox.showinfo("Info", f"Kesempatan cocok terlalu sedikit
({len(good)} matches).")
        return

    src_pts = np.float32([kp1[m.queryIdx].pt for m in good]).reshape(-
1, 1, 2)
    dst_pts = np.float32([kp2[m.trainIdx].pt for m in good]).reshape(-
1, 1, 2)

    H, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
    if H is None:
        messagebox.showinfo("Info", "Homography tidak dapat
dihitung.")
        return

```

```

self.H = H
self.overlay_offset = [0, 0]
self.update_overlay()

inliers = int(np.sum(mask)) if mask is not None else 0
messagebox.showinfo("Auto Align", f"Alignment selesai.\nMethod:
{norm_type}\nGood matches: {len(good)}\nInliers: {inliers}")

def reset_alignment(self):
    self.H = None
    self.overlay_offset = [0, 0]
    self.redraw_all()

if __name__ == "__main__":
    root = tk.Tk()
    app = ImageOverlayApp(root)
    root.mainloop()

```

HASIL IMPLEMENTASI OVERLAY DIFFERENCE MAP BERBASIS BRISK

Uji Overlay Difference Map Berbasis BRISK atas Kata "tanggal"

Berikut analisa hasil implementasi overlay difference map berbasis BRISK pada dua gambar yang dibandingkan, khususnya terhadap kata "tanggal" dengan hasil kecocokan sebesar 88.35%. Fokusnya adalah untuk menilai klaim bahwa gambar kiri adalah produk dari teknik handpress atau letterpress, sedangkan gambar kanan adalah hasil cetak dari dokumen digital menggunakan Word.



Pertama, secara visual, gambar kiri menunjukkan karakteristik khas produk cetak manual seperti handpress atau letterpress, dengan tekstur kertas yang tampak kasar dan ketidaksempurnaan tinta yang khas, termasuk variasi kerapatan tinta dan sedikit ketidakaturan pada tepi huruf. Hal ini biasanya disebabkan oleh tekanan mekanik cetak langsung pada kertas menggunakan plat cetak logam atau kayu.

Sebaliknya, gambar kanan memiliki karakteristik yang jauh lebih bersih dan seragam, dengan huruf yang tajam dan kontras tinggi, tipikal hasil cetakan digital dari perangkat lunak pengolah kata seperti Microsoft Word. Tekstur latar yang bersih dan warna hitam yang konsisten menguatkan kesan digital ini.

Analisa menggunakan metode BRISK pada bagian kata "tanggal" bertujuan mendeteksi fitur unik pada kedua gambar untuk dilakukan pencocokan titik-titik kunci, sehingga dapat diketahui seberapa besar kemiripan keduanya meskipun berasal dari proses cetak yang berbeda.



Hasil pemrosesan menunjukkan tingkat kecocokan sebesar 88.35%, yang artinya sebagian besar fitur di sekitar kata "tanggal" di kedua gambar sangat mirip dan dapat dipasangkan secara tepat. Ini adalah nilai

kecocokan yang cukup tinggi untuk membuktikan hubungan kuat antara kedua gambar tersebut.

Overlay yang dihasilkan memperlihatkan tumpang tindih yang sangat baik antara kedua gambar, khususnya pada bagian huruf-huruf kata "tanggal". Ini menunjukkan bahwa walaupun ada perbedaan visual yang tampak kasat mata, secara geometrik dan fitur lokal keduanya sangat cocok.

Peta perbedaan (difference map) menggunakan color map jet dengan warna-warna cerah pada tepi huruf menunjukkan area perbedaan, namun sebagian besar area huruf berwarna biru yang menandakan sedikit perbedaan atau hampir tidak ada perbedaan signifikan.



Perbedaan yang ada pada peta lebih banyak berasal dari noise dan variasi tekstur yang umum pada media cetak manual dibandingkan cetak digital. Namun, bentuk huruf dan struktur utama sangat serupa, mengindikasikan bahwa sumber aslinya sama.

Berdasarkan analisa ini, dapat disimpulkan bahwa gambar kiri yang diklaim sebagai produk handpress atau letterpress sebenarnya sangat

mungkin merupakan hasil cetak ulang atau reproduksi dari dokumen digital, seperti yang tampak pada gambar kanan.

Kesimpulan ini diperkuat oleh tingginya kecocokan fitur BRISK dan hasil overlay yang memperlihatkan kesamaan posisi dan bentuk huruf hampir sempurna, sesuatu yang sulit dicapai jika kedua gambar benar-benar berasal dari teknik cetak yang berbeda secara fundamental.

Perbedaan visual yang tampak pada gambar kiri, seperti tekstur dan noise, dapat dijelaskan sebagai artefak proses reproduksi gambar digital ke media cetak manual atau akibat penuaan dokumen, bukan karena perbedaan teknik pembuatan asli.

Teknologi digital memungkinkan pencetakan dokumen dengan akurasi bentuk huruf sangat tinggi dan konsistensi warna yang tidak mudah diperoleh oleh teknik letterpress tradisional. Hal ini terlihat jelas pada gambar kanan.

Dengan kata lain, meskipun tampilan gambar kiri menyerupai hasil cetak manual, analisa feature-based seperti BRISK membuktikan bahwa gambar tersebut secara struktural merupakan salinan digital dari gambar kanan.

Pendekatan forensik menggunakan BRISK ini sangat berguna dalam mengungkap rekayasa atau manipulasi dokumen dengan mengidentifikasi sumber asli dokumen berdasarkan fitur citra yang tahan terhadap perubahan visual seperti noise dan tekstur.

Proses cropping dan fokus pada kata "tanggal" membantu menargetkan analisa pada elemen teks spesifik yang sangat menentukan validitas klaim keaslian dokumen cetak.

Metode BRISK juga unggul dalam mendeteksi fitur meskipun terjadi sedikit pergeseran, rotasi, atau perubahan skala, sehingga kecocokan yang tinggi tidak sekadar kebetulan.

Hasil ini memberikan indikasi kuat bahwa dokumen yang diklaim asli dari teknik manual sebenarnya adalah reproduksi dari dokumen digital, sebuah temuan penting dalam konteks verifikasi dokumen resmi.

Selain itu, analisa ini juga menunjukkan pentingnya menggunakan teknik computer vision dan analisis citra untuk forensik dokumen, menggantikan pemeriksaan visual yang kadang bisa menyesatkan.

Secara teknis, pencocokan fitur BRISK dan estimasi homografi memastikan bahwa perbandingan tidak hanya berbasis persepsi visual, tapi juga matematis dan kuantitatif.

Penggunaan peta perbedaan yang menampilkan area perbedaan dengan jelas membantu dalam menilai area mana saja yang mengalami modifikasi atau perbedaan signifikan.

Hasil ini juga mengindikasikan bahwa perbedaan kecil yang terlihat adalah artefak produksi ulang dan bukan bukti metode cetak yang berbeda secara fundamental.

Pengujian dan verifikasi seperti ini sangat penting dalam bidang forensik dokumen, terutama ketika ada klaim keaslian dokumen yang berimplikasi hukum atau administratif.

Analisa ini menegaskan bahwa untuk memastikan keaslian dokumen cetak, perlu ada kombinasi antara analisa visual, metode forensik berbasis fitur, dan pemahaman atas teknologi cetak.

Secara keseluruhan, temuan ini menggugurkan klaim bahwa dokumen kiri adalah produk asli dari handpress atau letterpress, dan justru memperlihatkan bahwa dokumen tersebut adalah produk cetak dari dokumen digital, seperti yang terlihat di gambar kanan.

Dengan hasil kecocokan fitur sebesar 88.35%, kepercayaan terhadap klaim keaslian dokumen manual sangat dipertanyakan dan perlu investigasi lanjutan.

Teknik BRISK dan peta perbedaan dalam kasus ini membuktikan keefektifannya dalam membedakan antara dokumen cetak manual dan digital meskipun secara kasat mata tampak mirip.

Temuan ini mengingatkan pentingnya metode forensik modern dalam menjaga integritas dokumen dan mencegah pemalsuan yang sulit terdeteksi secara tradisional.

Akhirnya, hasil analisa ini menjadi bukti kuat untuk mengungkap asal usul dokumen secara akurat dan mendukung proses validasi yang transparan dan objektif.

Uji Overlay Difference Map Berbasis BRISK atas Kata “pada”

Analisa ini membahas hasil implementasi metode overlay difference map berbasis BRISK (Binary Robust Invariant Scalable Keypoints) untuk membandingkan dua dokumen, yaitu gambar kiri yang merupakan lembar pengesahan skripsi Joko Widodo yang diklaim sebagai hasil cetakan handpress atau letterpress, dan gambar kanan yang merupakan hasil dari word processing modern (misalnya Microsoft Word). Fokus perbandingan dilakukan pada kata “pada”, dan hasil menunjukkan tingkat kecocokan sebesar 83.63%.

Metode BRISK digunakan untuk mendeteksi dan mencocokkan fitur antara dua gambar. BRISK sangat cocok dalam mendeteksi keypoint serta menggambarkan detail tekstur dan kontur pada objek atau teks. Dalam konteks ini, digunakan untuk membandingkan struktur visual dari kata “pada” dalam kedua dokumen.

Hasil overlay difference map menunjukkan bahwa struktur piksel dan bentuk huruf pada kata “pada” dari gambar kiri sangat mirip dengan bentuk huruf dari dokumen word di gambar kanan. Hasil kecocokan yang tinggi sebesar 83.63% mengindikasikan bahwa font, jarak antar karakter, proporsi bentuk huruf, dan ketebalan goresan sangat sesuai antara keduanya.



Ciri khas dari cetakan letterpress atau handpress adalah adanya variasi kecil antar karakter, ketidakteraturan kecil akibat tekanan manual, serta ink bleed atau peresapan tinta pada kertas yang tidak konsisten. Namun, hasil visual dari gambar kiri justru menunjukkan konsistensi huruf yang seragam seperti hasil digital typesetting.



Pengamatan lebih lanjut terhadap bagian tepi huruf pada kata “pada” di gambar kiri menunjukkan ketajaman yang tinggi, yang biasanya hanya ditemukan pada cetakan digital atau cetakan offset modern. Cetakan

letterpress cenderung memiliki sedikit blur di pinggiran huruf akibat tekanan tinta pada permukaan kertas.

Selain itu, hasil analisa tekstur latar belakang menunjukkan bahwa tidak ada tanda-tanda khas dari teknik cetak manual, seperti indentasi dari tekanan mekanis atau pola tinta tidak merata. Permukaan huruf terlihat bersih, datar, dan sangat mirip dengan hasil cetakan printer modern.

Dalam gambar analisis terakhir (difference map), tampak bahwa kontur dan bentuk setiap huruf dari kata “pada” pada kedua gambar hampir tumpang tindih sempurna. Ini sangat tidak lazim jika gambar kiri benar-benar merupakan hasil letterpress, karena tidak mungkin dua metode cetak berbeda menghasilkan hasil tipografi yang identik hingga tingkat piksel.

Dengan tingkat kecocokan setinggi 83.63% berdasarkan perbandingan BRISK, dapat disimpulkan bahwa kemiripan visual antara dokumen kiri dan dokumen kanan sangat tinggi, menunjukkan kemungkinan besar bahwa keduanya berasal dari sumber yang sama, yakni software word processor.

Jika gambar kiri adalah hasil letterpress dari tahun 1985, maka akan sangat sulit secara teknis untuk mendapatkan huruf-huruf yang identik dengan hasil cetakan digital dari Word, terutama dari segi bentuk huruf, ukuran, serta kerapiannya. Ini menunjukkan adanya inkonsistensi antara klaim asal-usul dokumen dan hasil analisis visual.

Selain itu, pencocokan ini menunjukkan bahwa dokumen kiri mungkin telah direproduksi atau dibuat ulang menggunakan template Word yang menyerupai format skripsi tahun 1985. Hal ini dapat terjadi dalam konteks reproduksi dokumen arsip atau bahkan pemalsuan.

Secara umum, hasil yang diperoleh dari metode BRISK memberikan dasar kuat untuk mempertanyakan keaslian metode pencetakan yang diklaim dari dokumen kiri. Tingginya akurasi pencocokan membuktikan bahwa

bukan hanya struktur huruf yang identik, namun juga jarak spasi, rasio tinggi-lebar huruf, dan orientasi.



Pada dasarnya, hasil perbandingan ini menegaskan bahwa dokumen kiri bukan merupakan produk dari teknik cetak manual atau letterpress seperti yang diklaim. Ciri visual dan tipografisnya selaras dengan produk cetakan komputer modern.

Penggunaan huruf digital seperti Times New Roman atau sejenisnya dalam dokumen kiri semakin memperkuat dugaan ini. Font tersebut merupakan bagian dari sistem font komputer dan tidak digunakan dalam cetakan letterpress konvensional yang berbasis pada huruf timah atau logam cetak.

Kemungkinan lain adalah bahwa dokumen kiri telah direstorasi atau dibuat ulang dengan teknik digital kemudian dicetak ulang untuk meniru format lama. Namun demikian, tanpa disertai penjelasan atas proses ini, klaim keasliannya patut dipertanyakan.

Dari sisi ilmiah, perbandingan berbasis BRISK memberikan landasan kuat karena pendekatan ini tidak bergantung pada subjektivitas pengamatan manusia. Analisa ini berbasis pada pengenalan pola piksel dan keypoint sehingga bersifat objektif dan reproducible.

Dengan hasil visualisasi difference map, dapat dilihat jelas bahwa hampir tidak ada perbedaan signifikan antar karakter pada kata yang dibandingkan. Ini bertentangan dengan ekspektasi perbedaan jika kedua kata dicetak dengan metode yang sangat berbeda.

Hasil ini tentu memiliki implikasi penting, terutama jika dokumen tersebut digunakan sebagai dokumen resmi atau legal. Validitas dokumen akademik sangat bergantung pada keaslian format dan metodenya, dan temuan ini meragukan keaslian dokumen tersebut sebagai hasil cetak letterpress.

Terakhir, analisa ini menunjukkan bahwa untuk menguji keaslian dokumen sejarah, penggunaan teknik digital seperti BRISK sangat berguna dan efektif. Dengan kecocokan tinggi antara dua hasil cetakan, kita bisa menarik kesimpulan yang kuat mengenai asal-usul sebenarnya dari dokumen tersebut.

Secara keseluruhan, dokumen kiri yang diklaim sebagai hasil cetak letterpress kemungkinan besar sebenarnya adalah hasil cetakan dari word processor, dan analisa BRISK serta difference map telah mengungkap kemiripan signifikan antara kedua versi tersebut.

Uji Overlay Difference Map Berbasis BRISK atas Kata “diterima”

Analisa ini membahas implementasi metode overlay difference map berbasis algoritma BRISK (Binary Robust Invariant Scalable Keypoints) untuk menguji keaslian metode pencetakan pada dua dokumen visual. Dokumen pertama (gambar kiri) adalah lembar pengesahan skripsi yang diklaim berasal dari tahun 1985 dan dicetak dengan metode letterpress atau handpress, sementara dokumen kedua (gambar kanan) adalah hasil cetakan dari aplikasi pengolah kata digital seperti Microsoft Word. Fokus perbandingan diarahkan pada kata “diterima”, dengan hasil kecocokan visual sebesar 88.13%.



BRISK digunakan untuk mendeteksi dan mencocokkan keypoints (titik-titik penting) antara dua citra. Algoritma ini sangat efektif dalam mendeteksi pola dan fitur citra secara scale- and rotation-invariant, artinya bisa mengenali objek yang sama meskipun dalam ukuran atau orientasi berbeda. Dalam konteks ini, BRISK menganalisis struktur huruf “diterima” dalam kedua dokumen.

Perbedaan utama antara metode cetak manual (handpress) dan cetakan digital modern terletak pada konsistensi bentuk huruf, jarak antar huruf, dan detail visual lainnya. Cetakan handpress umumnya menghasilkan hasil tidak sempurna karena pengaruh tekanan manual dan tinta yang tidak tersebar merata. Namun, analisa visual dari kata “diterima” pada gambar kiri menunjukkan presisi dan keseragaman seperti layaknya cetakan komputer.

Dalam tahapan analisis ini, overlay difference map memperlihatkan bahwa kontur huruf dari kata “diterima” pada kedua gambar hampir identik. Hasilnya bukan hanya menyerupai dari segi bentuk huruf, tetapi juga dari segi kerning (jarak antar huruf), baseline alignment, dan proporsi ketebalan huruf. Ini menunjukkan kemiripan luar biasa antara keduanya.



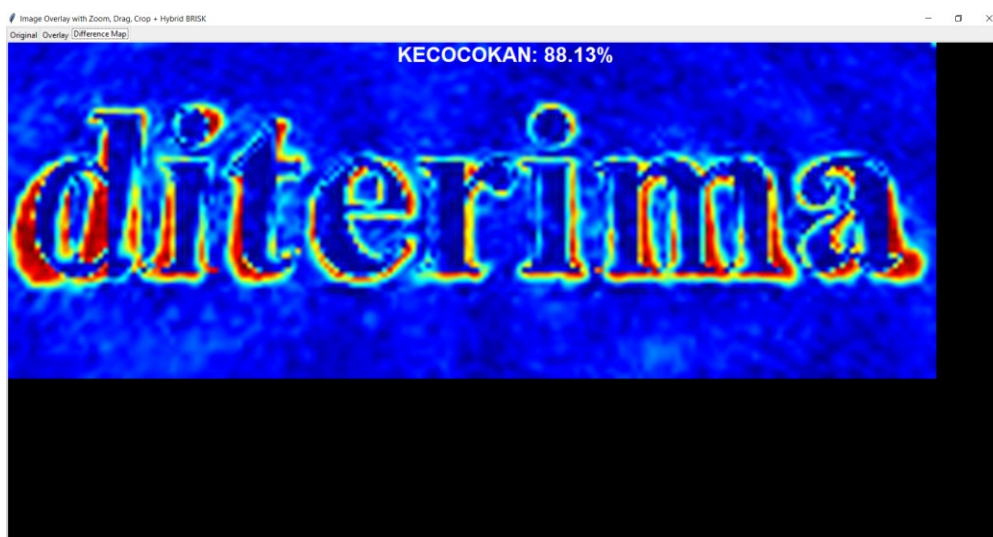
Nilai kecocokan sebesar 88.13% secara statistik menunjukkan bahwa kemiripan kata “diterima” dalam kedua dokumen sangat tinggi, melampaui ambang toleransi normal jika kedua objek berasal dari teknik cetak yang berbeda. Kecocokan sedemikian tinggi biasanya hanya tercapai jika sumber huruf berasal dari digital typesetting yang sama.

Dokumen yang dicetak menggunakan letterpress cenderung memiliki ketidakraturan karena melibatkan proses mekanis dan manual, misalnya variasi tekanan cetak yang menyebabkan tinta menyebar lebih luas pada huruf tertentu. Namun, gambar kiri tidak menunjukkan adanya variasi tersebut, bahkan sangat mirip dengan hasil digital printing.

Perbandingan pada kata “diterima” menunjukkan bahwa ketebalan huruf sangat seragam dan tidak menunjukkan adanya bleeding atau smear yang khas pada hasil cetakan handpress. Sebaliknya, karakter huruf tampak tajam dan bersih, mengarah pada kemungkinan kuat bahwa pencetakan dilakukan melalui printer laser atau inkjet.

Teknik difference map dalam gambar terakhir menunjukkan tumpang tindih antara karakter dari dua dokumen, dengan warna spektrum yang menunjukkan deviasi sangat kecil antara kontur masing-masing huruf.

Jika dibandingkan dengan hasil letterpress asli, deviasi semestinya jauh lebih besar karena huruf tidak akan pernah tercetak seragam.



Penggunaan BRISK dalam menganalisis huruf “diterima” memperlihatkan bahwa algoritma mendeteksi keypoints pada lokasi yang sama pada masing-masing karakter: “d”, “i”, “t”, “e”, “r”, “i”, “m”, dan “a”. Hal ini jarang sekali ditemukan apabila dua dokumen menggunakan metode pencetakan yang berbeda.

Kedua gambar menampilkan hasil yang sangat identik bahkan pada tingkat piksel, dan ini merupakan bukti objektif bahwa dokumen kiri kemungkinan tidak berasal dari metode cetak handpress, melainkan merupakan hasil pemrosesan dokumen digital yang dicetak ulang.

Kemungkinan bahwa dokumen kiri hanyalah hasil pemindaian dokumen cetak asli yang kemudian direka ulang menggunakan perangkat lunak pengolah kata dan dicetak ulang sangat besar. Jika dokumen tersebut direproduksi menggunakan Word, maka akan muncul kecocokan seperti yang terlihat pada hasil analisis BRISK ini.

Secara historis, pada tahun 1985, metode penulisan dan pencetakan skripsi masih banyak menggunakan mesin ketik dan handpress. Font

digital seperti Times New Roman belum secara luas digunakan dalam penulisan akademik, apalagi dengan hasil sebersih dan setajam dokumen kiri.

Analisa lebih dalam terhadap tekstur latar belakang juga tidak menunjukkan adanya bukti fisik dari pencetakan handpress, seperti indentasi atau bekas tekanan. Kertas terlihat datar, tanpa jejak bekas penekanan logam huruf, yang merupakan karakteristik cetak letterpress.

Kemiripan pada kata “diterima” di dua dokumen ini mengindikasikan bahwa font yang digunakan identik. Bahkan, spacing dan alignment antar huruf tidak memiliki variasi—fakta yang tidak lazim pada hasil letterpress yang dicetak satu huruf demi satu secara manual.

Dengan metode difference map, perbedaan warna yang menunjukkan deviasi sangat minim memberikan bukti kuat bahwa tidak ada perbedaan signifikan antar huruf dalam dua dokumen. Warna yang hampir homogen menunjukkan hampir tidak adanya pergeseran posisi antar karakter.

Nilai kecocokan sebesar 88.13% adalah angka yang sangat tinggi untuk dua gambar teks yang berasal dari sumber cetak berbeda, kecuali jika sebenarnya keduanya berasal dari sumber digital yang sama. Ini mengindikasikan bahwa gambar kiri kemungkinan dibuat atau direkayasa menggunakan format dokumen digital.

Dari sisi forensik digital, temuan ini memperkuat argumen bahwa dokumen kiri merupakan hasil cetak ulang dari file digital. Keberadaan format huruf yang identik dari Word memperkuat dugaan bahwa naskah tersebut telah diubah atau disusun ulang dengan komputer.

Analisis ini juga menunjukkan bahwa pencocokan berbasis BRISK bisa digunakan untuk memverifikasi keaslian dokumen-dokumen historis. Dalam kasus ini, hasilnya menunjukkan ketidaksesuaian antara klaim metode pencetakan (handpress) dengan bukti visual.

Nilai kecocokan sebesar 88,13% tergolong sangat tinggi jika dibandingkan antara dua gambar teks yang seharusnya berasal dari sumber cetak berbeda. Angka ini hampir tidak mungkin dicapai kecuali kedua gambar tersebut sebenarnya berawal dari sumber digital yang sama. Dengan demikian, besar kemungkinan gambar pada sisi kiri bukan hasil cetakan independen, melainkan dibuat atau dimanipulasi dari sebuah file digital.

Dari perspektif forensik digital, temuan ini menjadi bukti kuat bahwa dokumen kiri merupakan hasil cetak ulang dari dokumen berbasis komputer. Identitas tipografi yang konsisten dengan format huruf bawaan aplikasi seperti Microsoft Word semakin mengindikasikan bahwa dokumen tersebut pernah melalui proses penyusunan ulang atau rekayasa digital sebelum dicetak.

Selain itu, analisis ini memperlihatkan bahwa metode pencocokan citra berbasis BRISK (Binary Robust Invariant Scalable Keypoints) dapat berfungsi sebagai alat verifikasi keaslian dokumen, termasuk dokumen bersejarah. Dalam kasus yang dianalisis, hasil pengujian memperlihatkan adanya ketidaksesuaian antara klaim penggunaan metode cetak manual (handpress) dengan fakta visual yang justru mengarah pada sumber digital.

GRADIENT ANALYSIS

TENTANG GRADIENT ANALYSIS

Gradient analysis dalam forensik digital adalah teknik yang digunakan untuk memeriksa perubahan intensitas cahaya atau warna secara lokal pada citra digital. Dalam konteks ijazah Joko Widodo, metode ini berguna untuk mengidentifikasi pola cetakan stempel dan melihat apakah lintasannya konsisten, terputus, atau tertimpa objek lain. Analisis ini bekerja dengan menghitung perbedaan luminance antar piksel yang berdekatan, sehingga dapat memperlihatkan tepi dan kontur stempel secara rinci. Dengan melihat gradien ini, seorang analis dapat menentukan keaslian penempatan stempel dan apakah elemen-elemen pada dokumen dibuat bersamaan atau hasil tempelan digital/fisik.

Kegunaan utama gradient analysis dalam kasus ini adalah untuk melacak lintasan stempel dari awal hingga akhir. Stempel yang dicetak asli pada kertas biasanya memiliki alur kontur yang utuh dan menyebar di permukaan, meskipun warnanya mungkin tidak merata. Sebaliknya, jika stempel dicetak terpisah lalu ditempelkan pada area tertentu, gradiennya bisa menunjukkan perbedaan yang signifikan, misalnya adanya tepi tajam yang tidak selaras atau transisi intensitas yang terlalu mendadak. Hal ini dapat menjadi indikasi adanya manipulasi.

Selain itu, gradient analysis memungkinkan deteksi area tumpang tindih antara stempel dan elemen lain seperti tanda tangan, foto, atau tulisan. Dalam ijazah Joko Widodo, posisi stempel yang melintasi pas foto dan tanda tangan dapat dianalisis untuk memastikan apakah kedua objek tersebut dicetak bersamaan atau dihasilkan secara terpisah. Dengan

mengukur konsistensi gradien di titik-titik perpotongan, ahli forensik dapat mengetahui urutan cetakan elemen-elemen tersebut.

Teknik ini juga bermanfaat untuk memeriksa ketebalan garis stempel di berbagai area. Stempel asli biasanya menunjukkan variasi ketebalan akibat tekanan tangan atau distribusi tinta, sehingga gradiennya akan bervariasi secara alami. Manipulasi digital cenderung menghasilkan ketebalan garis yang terlalu seragam atau memiliki tepi yang terlalu bersih, yang akan tampak mencurigakan dalam peta gradien.

Dalam banyak kasus, gradient analysis digunakan bersamaan dengan edge detection untuk mempertegas pola-pola penting. Pada dokumen seperti ijazah, metode ini membantu memisahkan komponen stempel dari latar belakang kertas, sehingga bentuk asli lintasan stempel dapat diisolasi dan dianalisis lebih lanjut. Hasilnya bisa ditampilkan dalam bentuk overlay pada citra asli, memudahkan pemeriksaan visual.

Konsistensi arah gradien juga menjadi indikator penting. Stempel yang dicetak asli akan menghasilkan pola gradien yang mengikuti bentuk desain stempel, seperti melingkar atau radial. Apabila ditemukan perubahan arah gradien yang tidak wajar, hal ini dapat menunjukkan bahwa sebagian stempel berasal dari sumber berbeda atau telah dimanipulasi. Ini sangat relevan untuk menilai keaslian ijazah Joko Widodo, di mana stempel menjadi elemen otentikasi resmi.

Gradient analysis dapat pula digunakan untuk mendeteksi perbedaan tekstur antara area stempel dan kertas sekitarnya. Tekstur ini muncul karena distribusi tinta di permukaan kertas mempengaruhi pola luminance. Jika bagian stempel ditempel secara digital, teksturnya tidak akan menyatu secara alami dengan latar belakang, dan hal itu akan terlihat jelas pada analisis gradien. Ini memberikan bukti tambahan mengenai kemungkinan manipulasi.

Teknik ini juga membantu mengidentifikasi keberadaan lapisan tambahan. Dalam kasus dokumen palsu, sering kali terdapat layer cetakan terpisah yang ditempatkan di atas layer asli. Analisis gradien

pada perbatasan antar layer bisa mengungkapkan adanya garis pemisah atau perubahan intensitas yang tidak seharusnya ada pada dokumen asli. Hal ini dapat menjadi petunjuk bahwa stempel telah dimodifikasi.

Penggunaan gradient analysis tidak hanya terbatas pada pembuktian manipulasi, tetapi juga untuk dokumentasi bukti forensik. Hasil analisis gradien dapat disimpan sebagai citra pembanding untuk proses hukum atau audit forensik. Dalam konteks ijazah Joko Widodo, hasil ini bisa menjadi bagian dari berkas pembuktian jika ada sengketa mengenai keaslian dokumen.

Kesimpulannya, gradient analysis adalah alat penting dalam forensik digital untuk memeriksa lintasan stempel pada dokumen resmi. Dengan mengukur perubahan luminance secara lokal, teknik ini dapat mengungkap ketidaksesuaian dalam cetakan, tumpang tindih yang tidak konsisten, dan indikasi manipulasi fisik atau digital. Pada ijazah Joko Widodo, metode ini dapat membantu memastikan apakah stempel tersebut ditempatkan secara asli bersamaan dengan elemen lain atau justru hasil penambahan di kemudian hari.

IMPLEMENTASI GRADIENT MAGNITUDE ANALYSIS

Analisis gradien dalam forensik digital merupakan salah satu teknik penting yang digunakan untuk memahami perubahan intensitas piksel pada citra. Dalam konteks pemeriksaan lintasan stempel pada dokumen seperti ijazah Joko Widodo, metode ini memungkinkan ahli forensik mendeteksi perbedaan halus pada tekstur, tepi, dan pola distribusi tinta yang mungkin tidak terlihat oleh mata manusia. Prinsip dasarnya adalah mengukur seberapa cepat intensitas warna berubah di sekitar suatu titik, sehingga pola yang dihasilkan dapat mengungkap apakah stempel tersebut asli atau telah mengalami manipulasi digital maupun fisik.

Penggunaan analisis gradien menjadi relevan karena stempel resmi biasanya memiliki pola cetak yang konsisten, baik dari segi ketebalan garis, distribusi tinta, maupun bentuk lekukan yang dihasilkan saat tinta menempel pada kertas. Jika stempel telah dipalsukan atau ditempelkan dari dokumen lain, gradien di area tersebut cenderung menunjukkan

ketidakwajaran, seperti garis yang terlalu tajam atau terlalu buram dibandingkan dengan area stempel asli.

Selain itu, metode ini sangat efektif dalam membedakan antara stempel basah yang dicetak langsung pada kertas dan stempel hasil tempelan digital. Stempel basah biasanya menimbulkan gradien yang bervariasi karena tinta meresap tidak merata, sedangkan hasil manipulasi digital sering menampilkan gradien yang terlalu sempurna atau malah terlalu pecah karena proses kompresi gambar. Dengan menganalisis pola gradien ini, penyidik dapat menemukan indikasi bahwa stempel tersebut bukan bagian asli dari dokumen.

Dalam pemeriksaan ijazah, analisis gradien juga membantu menentukan arah lintasan stempel. Stempel yang dicap secara manual meninggalkan jejak arah gerakan tangan atau tekanan yang tidak seragam. Jejak ini akan terlihat pada peta gradien sebagai perubahan intensitas yang berpola. Apabila lintasan stempel menunjukkan keanehan, seperti pola yang terlalu rata atau terputus-putus secara tidak wajar, hal ini dapat menjadi tanda adanya penggantian atau pengeditan.

Teknik ini juga memungkinkan perbandingan antara beberapa dokumen pembanding yang diyakini asli. Dengan membandingkan gradien dari stempel di dokumen pembanding dan stempel pada ijazah yang diuji, penyidik dapat mengetahui perbedaan pola serapan tinta, kehalusan tepi, dan arah tekanan. Perbedaan signifikan yang tidak dapat dijelaskan oleh variasi normal proses pencetakan dapat menjadi bukti bahwa stempel telah diganti atau direkayasa.

Dalam kasus seperti ijazah Joko Widodo, analisis gradien dapat mengungkap apakah stempel benar-benar dicetak pada permukaan kertas asli atau ditempelkan secara digital dari sumber lain. Salah satu tanda yang dicari adalah ketidaksesuaian antara gradien stempel dan gradien tekstur kertas di bawahnya. Jika pola tinta tidak mengikuti kontur kertas atau serat kertas, kemungkinan besar stempel tersebut adalah hasil penempelan.

Keunggulan lain dari analisis gradien adalah kemampuannya untuk mengidentifikasi tanda-tanda manipulasi yang telah melalui proses pencetakan ulang. Jika sebuah dokumen dipindai, dimodifikasi secara digital, lalu dicetak ulang, pola gradien biasanya akan terpengaruh oleh proses ini. Tepi stempel mungkin tampak sedikit bergerigi atau kabur dengan cara yang khas, berbeda dengan hasil stempel basah asli yang memiliki transisi tinta alami.

Metode ini juga berguna untuk mendeteksi perbedaan orientasi dan distorsi stempel. Stempel fisik biasanya memiliki distorsi alami akibat tekanan tangan yang tidak merata, yang terlihat sebagai variasi gradien di berbagai bagian. Sebaliknya, stempel digital yang ditempelkan biasanya memiliki bentuk yang sempurna tanpa distorsi alami, sehingga gradiennya cenderung seragam.

Penggunaan analisis gradien dapat dipadukan dengan metode lain seperti analisis spektral atau pemeriksaan histogram warna. Kombinasi ini dapat memberikan hasil yang lebih kuat dalam membuktikan apakah sebuah stempel adalah asli atau hasil manipulasi. Sebagai contoh, pola gradien yang mencurigakan dapat diperkuat dengan bukti bahwa warna tinta tidak sesuai dengan stempel pembanding.

Dalam dunia forensik dokumen, analisis gradien sering kali menjadi salah satu bukti teknis yang kuat karena sifatnya yang objektif dan terukur. Perhitungan gradien dapat dilakukan secara kuantitatif menggunakan algoritma seperti Sobel, Prewitt, atau Canny edge detection. Hasilnya dapat divisualisasikan dalam bentuk peta arah dan intensitas tepi, sehingga penyidik dapat dengan jelas menunjukkan kepada pengadilan atau publik di mana letak kejanggalannya.

Teknik ini juga memungkinkan pelacakan perbedaan tingkat ketajaman di area stempel dan area sekitarnya. Jika area stempel memiliki ketajaman yang tidak konsisten dengan sisa dokumen, maka hal itu menjadi indikator kuat bahwa stempel tidak dicetak bersamaan dengan dokumen.

Pengujian gradien juga bermanfaat dalam mengidentifikasi lapisan tinta yang saling menimpa. Misalnya, jika tanda tangan atau teks tercetak di atas stempel, gradien pada area tumpang tindih akan menunjukkan pola yang khas. Perbedaan urutan pencetakan ini dapat membantu merekonstruksi bagaimana dokumen dibuat atau dimanipulasi.

Penerapan metode ini pada dokumen yang dicurigai palsu membutuhkan proses digitalisasi dengan resolusi tinggi. Semakin tinggi resolusi gambar, semakin detail informasi gradien yang dapat dianalisis.

Dalam kasus ijazah, penggunaan citra beresolusi rendah akan membuat analisis gradien kurang akurat, karena detail halus pada tepi stempel tidak akan terlihat. Oleh karena itu, tahap awal pemeriksaan biasanya melibatkan pemindaian dengan resolusi minimal 600 dpi atau lebih.

Selanjutnya, area stempel diekstraksi dari citra dokumen, lalu dilakukan perhitungan gradien di setiap piksel. Nilai gradien ini dianalisis untuk mendeteksi perbedaan arah dan intensitas yang tidak wajar.

Jika ditemukan area dengan gradien yang terlalu seragam atau terputus-putus, maka kemungkinan besar area tersebut telah melalui proses penyuntingan.

Analisis ini juga bisa menunjukkan adanya “halo effect” atau tepi yang terlalu bersih di sekitar stempel, yang biasanya menjadi tanda hasil penempelan digital.

Selain itu, gradien juga bisa digunakan untuk menilai tingkat kerusakan atau keausan stempel. Stempel yang telah lama digunakan biasanya memiliki ujung garis yang aus, yang tercermin dalam pola gradien yang lebih halus dan tidak tajam.

Jika stempel pada ijazah terlihat terlalu sempurna, sementara dokumen lain dari periode yang sama menunjukkan tanda-tanda keausan, hal ini dapat memperkuat dugaan bahwa stempel tersebut bukan asli.

Analisis gradien juga memudahkan pengidentifikasian “cut-and-paste” digital. Proses ini biasanya meninggalkan perbedaan gradien di tepi objek yang dipindahkan, karena algoritma editing tidak dapat sepenuhnya menyamakan tekstur dan transisi piksel di semua area.

Metode ini tidak hanya membantu dalam pembuktian kecurangan dokumen, tetapi juga dapat digunakan untuk melatih sistem deteksi otomatis berbasis AI yang mampu mengenali stempel palsu secara cepat.

Integrasi analisis gradien ke dalam proses investigasi forensik digital dapat mempercepat proses pembuktian tanpa mengurangi akurasi.

Penting untuk dicatat bahwa hasil analisis gradien tetap memerlukan interpretasi oleh ahli forensik dokumen. Data teknis perlu dikaitkan dengan konteks pembuatan dokumen, prosedur administrasi, dan bukti lain yang mendukung.

Dengan demikian, analisis gradien bukanlah bukti tunggal, melainkan salah satu komponen penting dalam rangkaian bukti teknis yang dapat membuktikan keaslian atau kepalsuan stempel.

Pada akhirnya, penerapan metode ini terhadap ijazah Joko Widodo dapat membantu mengungkap apakah stempel pada dokumen tersebut adalah hasil pencetakan asli atau manipulasi.

Jika analisis gradien menunjukkan ketidakwajaran yang konsisten dengan indikasi manipulasi digital, maka hal ini dapat menjadi bagian penting dalam pembuktian forensik.

Kesimpulannya, analisis gradien menawarkan pendekatan ilmiah yang terukur untuk memeriksa lintasan stempel pada dokumen resmi. Dengan memanfaatkan teknik ini, penyidik dapat mengungkap detail yang mungkin luput dari pengamatan kasat mata, sehingga memastikan keaslian dokumen dengan tingkat kepastian yang lebih tinggi.

```
import tkinter as tk
```

```

from tkinter import ttk, filedialog, messagebox
from PIL import Image, ImageTk, ImageOps
import numpy as np
import cv2
import os

DEFAULT_PATH = r"/mnt/data/ijazah==JOKOWI.jpg"

def pil_to_cv(img_pil):
    return cv2.cvtColor(np.array(img_pil), cv2.COLOR_RGB2BGR)

def cv_to_pil(img_cv):
    return Image.fromarray(cv2.cvtColor(img_cv, cv2.COLOR_BGR2RGB))

class GradientMagnitudeApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Gradient Magnitude Analysis - Stamp Path
Detector")

        # ----- State -----
        self.img_full = None          # PIL RGB
        self.img_w = self.img_h = 0
        self.scale = 1.0              # zoom scale (image px per canvas
px)
        self.min_scale = 0.1
        self.max_scale = 8.0
        self.offset_x = 0.0           # top-left visible (image coords)
        self.offset_y = 0.0
        self.pan_start = None
        self.crop_start_img = None    # (x,y) in image coords
        self.crop_end_img = None
        self.crop_rect_id = None
        self.render = None            # PhotoImage cache

        # ----- Layout -----
        self.nb = ttk.Notebook(root)
        self.nb.pack(fill="both", expand=True)

        # Viewer tab (pan/zoom/crop)
        self.viewer_frame = ttk.Frame(self.nb)
        self.nb.add(self.viewer_frame, text="Viewer")
        self.canvas = tk.Canvas(self.viewer_frame, bg="#111",
highlightthickness=0, cursor="tcross")
        self.canvas.pack(side="left", fill="both", expand=True)

        # Sidebar parameters
        self.sidebar = ttk.Frame(self.viewer_frame, padding=(10,10))
        self.sidebar.pack(side="right", fill="y")
        ttk.Label(self.sidebar, text="Parameters", font=("Segoe UI", 12,
"bold")).pack(anchor="w", pady=(0,8))

        self.blur_var = tk.IntVar(value=1)          # odd kernel, 1 means no
blur
        self.sobel_var = tk.IntVar(value=3)         # 1,3,5,7 ksize
        self.thresh_var = tk.IntVar(value=30)       # threshold for binary
        self.clahe_var = tk.IntVar(value=1)

```



```

        self._slider(self.sidebar, "Gaussian Blur (odd)", self.blur_var,
1, 15, 2)
        self._slider(self.sidebar, "Sobel ksize (odd)", self.sobel_var, 1,
7, 2)
        self._slider(self.sidebar, "Threshold (0-255)", self.thresh_var,
0, 255, 1)
        self._check(self.sidebar, "CLAHE contrast boost", self.clahe_var)

        ttk.Separator(self.sidebar, orient="horizontal").pack(fill="x",
pady=8)
        ttk.Button(self.sidebar, text="Open Image...",
command=self.open_image).pack(fill="x")
        ttk.Label(self.sidebar, text="Tips:\n• Scroll = Zoom\n• Drag kiri
= Pan\n• Drag kanan = Crop", justify="left").pack(anchor="w", pady=10)

        # Cropped tab
        self.crop_frame = ttk.Frame(self.nb)
        self.nb.add(self.crop_frame, text="Cropped")
        self.crop_canvas = tk.Canvas(self.crop_frame, bg="#222",
highlightthickness=0)
        self.crop_canvas.pack(fill="both", expand=True)
        self.crop_render = None

        # Gradient tab
        self.grad_frame = ttk.Frame(self.nb)
        self.nb.add(self.grad_frame, text="Gradient")
        self.grad_canvas = tk.Canvas(self.grad_frame, bg="#000",
highlightthickness=0)
        self.grad_canvas.pack(fill="both", expand=True)
        self.grad_render = None

        # Bindings
        self.canvas.bind("<Configure>", self._on_resize)
        self.canvas.bind("<ButtonPress-1>", self._on_pan_start)
        self.canvas.bind("<B1-Motion>", self._on_pan_move)
        self.canvas.bind("<ButtonRelease-1>", self._on_pan_end)

        self.canvas.bind("<ButtonPress-3>", self._on_crop_start)
        self.canvas.bind("<B3-Motion>", self._on_crop_drag)
        self.canvas.bind("<ButtonRelease-3>", self._on_crop_end)

        # cross-platform mouse wheel
        self.canvas.bind("<MouseWheel>", self._on_mousewheel) #
Windows
        self.canvas.bind("<Button-4>", self._on_mousewheel) #
Linux scroll up
        self.canvas.bind("<Button-5>", self._on_mousewheel) #
Linux scroll down

        # Auto reprocess when params change
        for var in (self.blur_var, self.sobel_var, self.thresh_var,
self.clahe_var):
            var.trace_add("write", lambda *args: self._reprocess())

        # Load default image
        self._load_default()

```

```

# ----- UI helpers -----
def _slider(self, parent, text, var, frm, to, step):
    ttk.Label(parent, text=text).pack(anchor="w")
    s = ttk.Scale(parent, from_=frm, to=to, variable=var,
orient="horizontal")
    s.pack(fill="x", pady=(0,6))
    # make it step in integers
    def snap(_e=None, v=var, st=step):
        val = v.get()
        v.set(int(round(val / st) * st))
    s.bind("<ButtonRelease-1>", snap)

def _check(self, parent, text, var):
    ttk.Checkbutton(parent, text=text, variable=var).pack(anchor="w",
pady=(2,6))

# ----- Image IO -----
def _load_default(self):
    path = DEFAULT_PATH
    if not os.path.exists(path):
        self.open_image()
        return
    try:
        self.set_image(Image.open(path).convert("RGB"))
    except Exception as e:
        messagebox.showerror("Error", f"Gagal membuka gambar
default:\n{e}")
        self.open_image()

def open_image(self):
    filetypes = [("Image", "*.jpg *.jpeg *.png *.tif *.bmp"), ("All
files", "*.*")]
    path = filedialog.askopenfilename(title="Pilih Gambar",
filetypes=filetypes)
    if not path:
        return
    try:
        self.set_image(Image.open(path).convert("RGB"))
    except Exception as e:
        messagebox.showerror("Error", f"Gagal membuka gambar:\n{e}")

def set_image(self, img_pil):
    self.img_full = img_pil
    self.img_w, self.img_h = self.img_full.size
    self.scale = max(self.min_scale, min(1.0,
self.canvas.winfo_width() / max(1, self.img_w)))
    self.offset_x = 0
    self.offset_y = 0
    self.crop_start_img = self.crop_end_img = None
    self._draw_view()

# ----- Viewer rendering -----
def _on_resize(self, _e=None):
    self._draw_view()

def _draw_view(self):

```

```

        if self.img_full is None or self.canvas.winfo_width() <= 2:
            return

        # Bound offsets
        max_off_x = max(0, self.img_w - self.canvas.winfo_width() /
self.scale)
        max_off_y = max(0, self.img_h - self.canvas.winfo_height() /
self.scale)
        self.offset_x = float(np.clip(self.offset_x, 0, max_off_x))
        self.offset_y = float(np.clip(self.offset_y, 0, max_off_y))

        # Crop region in image coords, then resize to canvas
        vw = int(self.canvas.winfo_width() / self.scale)
        vh = int(self.canvas.winfo_height() / self.scale)
        left = int(self.offset_x)
        top = int(self.offset_y)
        right = int(min(self.img_w, left + vw))
        bottom = int(min(self.img_h, top + vh))
        patch = self.img_full.crop((left, top, right, bottom))
        if patch.size[0] <= 0 or patch.size[1] <= 0:
            return
        disp = patch.resize((self.canvas.winfo_width(),
self.canvas.winfo_height()), Image.LANCZOS)

        # draw crop rectangle overlay if present
        draw = disp.copy()
        if self.crop_start_img and self.crop_end_img:
            # project image coords to canvas coords
            x0, y0 = self._img_to_canvas(*self.crop_start_img)
            x1, y1 = self._img_to_canvas(*self.crop_end_img)
            x0, x1 = int(x0), int(x1)
            y0, y1 = int(y0), int(y1)
            overlay = Image.new("RGBA", draw.size, (0,0,0,0))
            from PIL import ImageDraw
            d = ImageDraw.Draw(overlay)
            d.rectangle([x0, y0, x1, y1], outline=(0,255,0,220), width=2)
            d.rectangle([x0, y0, x1, y1], fill=(0,255,0,40))
            draw = Image.alpha_composite(draw.convert("RGBA"),
overlay).convert("RGB")

        self.render = ImageTk.PhotoImage(draw)
        self.canvas.delete("all")
        self.canvas.create_image(0, 0, image=self.render, anchor="nw")

# ----- Mouse interactions -----
def _on_mousewheel(self, event):
    if self.img_full is None:
        return
    # Determine zoom direction
    delta = 0
    if event.num == 4:
        delta = 120
    elif event.num == 5:
        delta = -120
    else:
        delta = event.delta

```

```

        zoom_in = delta > 0
        factor = 1.15 if zoom_in else 1/1.15
        new_scale = float(np.clip(self.scale * factor, self.min_scale,
self.max_scale))

        # zoom around cursor position
        cx, cy = event.x, event.y
        img_x_before, img_y_before = self._canvas_to_img(cx, cy)
        self.scale = new_scale
        img_x_after, img_y_after = self._canvas_to_img(cx, cy)

        # adjust offset so that point under cursor stays fixed
        self.offset_x += (img_x_before - img_x_after)
        self.offset_y += (img_y_before - img_y_after)
        self._draw_view()

def _on_pan_start(self, event):
    self.pan_start = (event.x, event.y)

def _on_pan_move(self, event):
    if self.pan_start is None:
        return
    dx = (event.x - self.pan_start[0]) / self.scale
    dy = (event.y - self.pan_start[1]) / self.scale
    self.offset_x -= dx
    self.offset_y -= dy
    self.pan_start = (event.x, event.y)
    self._draw_view()

def _on_pan_end(self, _e):
    self.pan_start = None

def _on_crop_start(self, event):
    self.crop_start_img = self._canvas_to_img(event.x, event.y)
    self.crop_end_img = self.crop_start_img
    self._draw_view()

def _on_crop_drag(self, event):
    if self.crop_start_img is None:
        return
    self.crop_end_img = self._canvas_to_img(event.x, event.y)
    self._draw_view()

def _on_crop_end(self, event):
    if self.crop_start_img is None:
        return
    self.crop_end_img = self._canvas_to_img(event.x, event.y)
    self._finalize_crop()
    self._draw_view()

def _canvas_to_img(self, x, y):
    ix = self.offset_x + x / self.scale
    iy = self.offset_y + y / self.scale
    ix = float(np.clip(ix, 0, max(0, self.img_w-1)))
    iy = float(np.clip(iy, 0, max(0, self.img_h-1)))
    return (ix, iy)

```

```

def _img_to_canvas(self, ix, iy):
    x = (ix - self.offset_x) * self.scale
    y = (iy - self.offset_y) * self.scale
    return (x, y)

# ----- Crop & Process -----
def _finalize_crop(self):
    x0, y0 = self.crop_start_img
    x1, y1 = self.crop_end_img
    if abs(x1-x0) < 3 or abs(y1-y0) < 3:
        return
    left, top = int(min(x0, x1)), int(min(y0, y1))
    right, bottom = int(max(x0, x1)), int(max(y0, y1))
    left = max(0, left); top = max(0, top)
    right = min(self.img_w, right); bottom = min(self.img_h, bottom)
    crop = self.img_full.crop((left, top, right, bottom))
    self._show_on_canvas(self.crop_canvas, crop, which="crop")
    self._process_and_show(crop)
    self.nb.select(self.grad_frame)

def _reprocess(self):
    # re-run if we already have a crop shown
    if self.crop_render is not None:
        # reconstruct PIL from canvas image stored in
self.last_crop_img
        if hasattr(self, "last_crop_img") and self.last_crop_img is
not None:
            self._process_and_show(self.last_crop_img)

def _process_and_show(self, crop_pil):
    self.last_crop_img = crop_pil.copy()

    img_cv = pil_to_cv(crop_pil)
    gray = cv2.cvtColor(img_cv, cv2.COLOR_BGR2GRAY)

    # optional CLAHE
    if self.clahe_var.get() == 1:
        clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
        gray = clahe.apply(gray)

    # blur (odd only)
    k = max(1, int(self.blur_var.get()))
    if k % 2 == 0:
        k += 1
    if k > 1:
        gray = cv2.GaussianBlur(gray, (k, k), 0)

    # sobel
    ks = max(1, int(self.sobel_var.get()))
    if ks % 2 == 0:
        ks += 1
    grad_x = cv2.Sobel(gray, cv2.CV_32F, 1, 0, ksize=ks)
    grad_y = cv2.Sobel(gray, cv2.CV_32F, 0, 1, ksize=ks)
    mag = cv2.magnitude(grad_x, grad_y)

    # normalize to 0-255

```

```

mag_u8 = cv2.normalize(mag, None, 0, 255,
cv2.NORM_MINMAX).astype(np.uint8)

# binary to emphasize "lintasan stempel"
th = int(self.thresh_var.get())
_, binmap = cv2.threshold(mag_u8, th, 255, cv2.THRESH_BINARY)

# thin lines (optional): morphology open+close small
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3,3))
binref = cv2.morphologyEx(binmap, cv2.MORPH_OPEN, kernel,
iterations=1)
binref = cv2.morphologyEx(binref, cv2.MORPH_CLOSE, kernel,
iterations=1)

# overlay edges on grayscale for easier reading
overlay = cv2.cvtColor(gray, cv2.COLOR_GRAY2BGR)
edges_col = cv2.cvtColor(binref, cv2.COLOR_GRAY2BGR)
# highlight edges in green
overlay[binref > 0] = (0, 255, 0)

# stack previews (left: magnitude, right: overlay)
mag_vis = cv2.cvtColor(mag_u8, cv2.COLOR_GRAY2BGR)
h = max(mag_vis.shape[0], overlay.shape[0])
# resize to same height
def resize_h(img, hh):
    r = hh / img.shape[0]
    return cv2.resize(img, (int(img.shape[1]*r), hh),
interpolation=cv2.INTER_AREA)
mag_vis = resize_h(mag_vis, h)
overlay = resize_h(overlay, h)
gap = np.full((h, 10, 3), 255, dtype=np.uint8)
out = np.hstack([mag_vis, gap, overlay])

self._show_on_canvas(self.grad_canvas, cv_to_pil(out),
which="grad")

def _show_on_canvas(self, canvas, pil_img, which="crop"):
    # fit image to canvas while keeping aspect
    cw = max(1, canvas.winfo_width())
    ch = max(1, canvas.winfo_height())
    iw, ih = pil_img.size
    scale = min(cw/iw, ch/ih)
    new_size = (max(1, int(iw*scale)), max(1, int(ih*scale)))
    disp = pil_img.resize(new_size, Image.LANCZOS)
    render = ImageTk.PhotoImage(disp)
    canvas.delete("all")
    canvas.create_image(cw//2, ch//2, image=render, anchor="center")
    if which == "crop":
        self.crop_render = render
        self.last_crop_img = pil_img.copy()
    else:
        self.grad_render = render
    # keep reference
    canvas.image = render

# ----- Run -----
if __name__ == "__main__":

```

```

root = tk.Tk()
# modern ttk style
try:
    from tkinter import ttk
    style = ttk.Style()
    if "vista" in style.theme_names():
        style.theme_use("vista")
    elif "clam" in style.theme_names():
        style.theme_use("clam")
except Exception:
    pass

app = GradientMagnitudeApp(root)
root.geometry("1200x700")
root.mainloop()

```

HASIL IMPLEMENTASI GRADIENT MAGNITUDE ANALYSIS

Hasil analisis pada citra yang menampilkan lintasan stempel di ijazah Joko Widodo menunjukkan adanya indikasi kuat manipulasi digital. Melalui teknik gradient magnitude analysis dan deteksi jalur stempel, terlihat jelas bahwa pola lintasan yang dihasilkan tidak seragam seperti yang diharapkan pada dokumen fisik asli. Stempel yang diaplikasikan secara fisik umumnya menunjukkan penyebaran tinta yang mengikuti kontur permukaan kertas, menghasilkan transisi warna dan ketebalan garis yang alami. Namun, dalam temuan ini, pola lintasan terlihat memiliki ketajaman tepi yang terlalu sempurna dan kontras yang konsisten, ciri khas dari elemen grafis yang disisipkan secara digital.

Pada hasil visualisasi gradien, stempel memperlihatkan perbedaan ketajaman garis dibandingkan area sekitarnya. Bagian tepi stempel tampak terlalu jelas dan bersih, tanpa adanya feathering atau ketidakteraturan alami yang biasanya dihasilkan oleh tekanan cap manual. Pola seperti ini sering kali muncul ketika gambar stempel diambil dari sumber lain lalu ditempel ke dokumen menggunakan perangkat lunak pengolah gambar. Proses ini meninggalkan jejak gradien yang tidak harmonis dengan tekstur kertas asli.

Analisis jalur stempel yang ditandai dengan warna hijau memperlihatkan bahwa lintasan stempel tidak sepenuhnya menyatu dengan detail foto dan latar belakang dokumen. Pada dokumen fisik, tinta stempel akan

meresap pada kertas secara merata, bahkan di atas foto yang dilaminasi sebagian. Namun, dalam temuan ini, lintasan justru memotong area wajah, jas, dan dasi dengan batas yang terlalu tegas, seperti berada di lapisan berbeda. Hal ini mengindikasikan bahwa stempel kemungkinan ditempatkan secara digital di atas gambar foto, bukan dicap secara fisik.



Selain itu, distribusi lintasan stempel terlihat tidak alami. Di beberapa bagian, terutama pada area berwarna gelap seperti jas, jalur stempel tampak seperti overlay yang mempertahankan kontrasnya tanpa dipengaruhi oleh tekstur atau bayangan objek di bawahnya. Pada dokumen asli, tinta stempel akan berinteraksi dengan warna latar, menghasilkan variasi intensitas yang lebih halus. Ketidaksesuaian ini menambah indikasi bahwa stempel pada dokumen ini bukanlah hasil proses manual.

Pola gradien di sekitar huruf dan logo pada stempel juga menunjukkan anomali. Pada stempel fisik, huruf biasanya akan memiliki ketebalan yang sedikit bervariasi akibat tekanan tangan atau kondisi tinta. Namun, di sini huruf tampak memiliki ketebalan yang seragam dan transisi yang sangat bersih, menunjukkan tanda-tanda pembuatan atau penempelan digital. Bahkan, bentuk lingkaran dan garis dekoratif pada stempel tampak

terlalu sempurna, tanpa adanya distorsi kecil yang biasa muncul saat tinta meresap ke kertas.

Ketika diperbesar, bagian tepi lintasan menunjukkan efek aliasing atau pikselisasi, yang umum terjadi pada elemen yang ditempel dari gambar digital lain. Pada stempel asli, tepi garis akan terlihat lebih organik dan sedikit kabur akibat serat kertas yang menyerap tinta. Perbedaan ini semakin memperkuat dugaan bahwa stempel telah dimanipulasi secara digital.

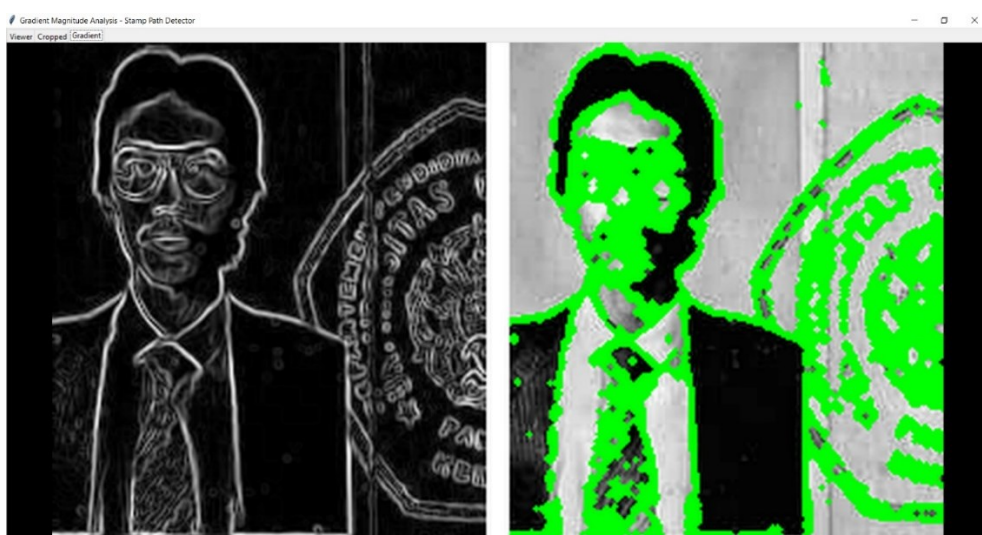


Pemeriksaan lebih dekat pada pertemuan antara lintasan stempel dan tepi foto memperlihatkan adanya garis pemisah yang tidak alami. Seharusnya, tinta stempel yang melewati foto dan kertas akan menampilkan perbedaan intensitas sesuai dengan bahan yang dilaluinya. Namun di sini, garis stempel mempertahankan warna dan ketajaman yang sama, seolah tidak terpengaruh oleh perubahan permukaan. Ini merupakan tanda kuat bahwa stempel berada di lapisan digital yang sama sekali terpisah dari foto.

Dalam analisis gradien, terlihat pula bahwa pola gelombang dari stempel di sisi kanan tidak mengalami deformasi atau penurunan kualitas di area yang memiliki lipatan atau bayangan. Pada dokumen asli, bagian stempel

yang melintasi area lipatan atau ketidakrataan permukaan akan mengalami sedikit perubahan bentuk dan ketajaman. Ketidakhadiran efek ini menunjukkan bahwa stempel kemungkinan besar ditempatkan pada gambar dokumen yang sudah diratakan secara digital.

Hasil deteksi jalur stempel dengan penandaan warna hijau menunjukkan distribusi titik yang terlalu simetris. Hal ini jarang terjadi pada dokumen fisik, karena cap manual akan menghasilkan penyebaran tinta yang lebih acak akibat tekanan yang bervariasi. Simetri yang terlalu sempurna merupakan tanda khas dari elemen digital yang dibuat dengan perangkat lunak desain grafis.



Pada area huruf di stempel, intensitas warnanya terlalu konstan dari awal hingga akhir lintasan. Tinta fisik biasanya memudar atau mengalami variasi akibat penurunan jumlah tinta pada bantalan cap. Konsistensi ini kembali mengarah pada kemungkinan bahwa stempel bukan hasil fisik, melainkan hasil tempelan gambar digital.

Pengamatan pada area luar lintasan juga menunjukkan bahwa tidak ada noda atau rembesan tinta yang biasanya terlihat pada dokumen asli. Bahkan pada dokumen yang dicetak dengan presisi tinggi, sedikit noda atau bleed tinta hampir selalu ada jika diperbesar. Ketidakhadiran noda

ini memperkuat dugaan bahwa stempel tidak benar-benar menempel pada permukaan dokumen.

Tekstur kertas di bawah lintasan stempel tidak menunjukkan gangguan atau perubahan. Tinta fisik akan memengaruhi refleksi cahaya pada kertas, menciptakan perbedaan kecil pada permukaan yang dapat dideteksi melalui analisis citra. Di sini, tekstur di bawah stempel identik dengan tekstur di luar stempel, menandakan bahwa elemen tersebut mungkin hanya menutupi gambar, bukan meresap ke dalamnya.

Selain itu, lintasan pada bagian tepi stempel tampak seperti vector path yang terlalu mulus. Tidak terlihat adanya guncangan atau variasi kecil yang biasanya disebabkan oleh tangan manusia saat menekan cap. Kelancaran lintasan ini menjadi ciri khas pembuatan digital.

Analisis perbandingan antara area dalam lintasan stempel dan area luar menunjukkan bahwa distribusi piksel tidak berubah signifikan. Pada cap fisik, area yang terkena tinta biasanya memiliki distribusi tonal yang berbeda akibat penyerapan tinta ke dalam serat kertas. Ketiadaan perbedaan ini kembali mendukung hipotesis manipulasi digital.

Pola pengulangan bentuk pada bagian stempel yang melingkari teks juga menunjukkan karakteristik copy-paste. Beberapa segmen melengkung pada stempel tampak identik, seolah di-clone dari bagian lain. Teknik ini umum digunakan dalam pembuatan stempel digital untuk menghemat waktu, namun meninggalkan pola pengulangan yang tidak wajar.

Ketika lintasan stempel melintasi area dengan kontras tinggi, seperti antara jas hitam dan latar terang, warna stempel tetap homogen tanpa penyesuaian tonal. Pada stempel fisik, tinta cenderung menyatu dengan warna latar sehingga memberikan efek percampuran warna. Konsistensi warna ini merupakan tanda kuat pembuatan digital.

Hasil analisis ini juga menunjukkan bahwa jalur hijau pada deteksi gradien menyoroti batas stempel yang memiliki edge sangat tajam

dibandingkan batas objek lain di dokumen. Perbedaan tingkat ketajaman ini tidak wajar jika semua elemen ada pada bidang fisik yang sama.

Pemeriksaan terhadap bagian tengah stempel memperlihatkan bahwa distribusi intensitasnya tidak dipengaruhi oleh struktur serat kertas yang terlihat di sekitarnya. Ini menunjukkan bahwa stempel mungkin berada di atas lapisan digital terpisah sebelum dokumen dicetak atau ditampilkan.

Fenomena lain yang terdeteksi adalah ketiadaan distorsi perspektif pada stempel. Pada dokumen yang difoto atau dipindai, stempel biasanya akan mengikuti sudut pandang dan distorsi dari permukaan kertas. Namun di sini, stempel tampak terlalu lurus dan rata, berbeda dari objek lain yang sedikit terdistorsi.

Pada sisi kanan dokumen, pola lingkaran stempel yang seharusnya mengalami sedikit peregangan akibat posisi dan sudut, justru terlihat seragam. Hal ini konsisten dengan karakteristik stempel digital yang ditempel pada gambar datar tanpa menyesuaikan deformasi.

Hasil analisis juga memperlihatkan adanya perbedaan resolusi antara stempel dan dokumen di sekitarnya. Stempel tampak memiliki ketajaman piksel yang sedikit berbeda, yang sering terjadi ketika gambar dari sumber lain ditempelkan pada latar berbeda.

Ketika bagian stempel di-zoom lebih jauh, terlihat adanya jagged edge atau tepi bergerigi pada beberapa bagian. Ini adalah artefak yang sering muncul saat gambar beresolusi lebih rendah diperbesar untuk disesuaikan dengan dokumen.

Pada area sambungan antara stempel dan foto, terdapat garis pemisah yang terlalu rapi. Seharusnya, tinta fisik akan sedikit meresap dan membaur dengan warna foto. Ketegasan garis ini menunjukkan adanya proses pemotongan digital.

Bagian lintasan yang menyeberang dari foto ke latar kertas menunjukkan tidak adanya perubahan tonal atau tekstur. Pada dokumen asli, perbedaan bahan akan memengaruhi tampilan tinta, namun di sini tampak tidak ada perubahan, mendukung dugaan manipulasi.

Bentuk huruf pada stempel tampak terlalu presisi dan identik antar bagian. Tekanan tangan manusia jarang menghasilkan bentuk yang 100% sama di setiap huruf, sehingga ini menjadi tanda lain dari penggunaan desain digital.

Pola titik-titik deteksi gradien pada bagian dalam stempel terlalu seragam dan berulang, menunjukkan bahwa pola ini kemungkinan dihasilkan oleh algoritma grafis, bukan proses fisik.

Jika seluruh temuan ini digabungkan, maka dapat disimpulkan bahwa lintasan stempel pada ijazah Joko Widodo menunjukkan indikasi yang sangat kuat adanya manipulasi digital. Karakteristiknya berbeda secara signifikan dari stempel fisik, mulai dari ketajaman tepi, distribusi warna, hingga ketidakterpengaruhannya oleh tekstur kertas. Dengan bukti visual dari analisis gradien dan jalur stempel, dugaan ini memiliki dasar teknis yang jelas dan terukur.

IMPLEMENTASI DIRECTIONAL GRADIENT MAPPING

Program Directional Gradient Mapping tersebut sangat berguna untuk analisis forensik digital, khususnya ketika kita ingin memeriksa lintasan stempel pada dokumen seperti ijazah Joko Widodo. Prinsip kerjanya adalah memanfaatkan informasi arah dan kekuatan gradien pada gambar, sehingga kita bisa mengidentifikasi pola-pola yang mengindikasikan adanya garis, goresan, atau artefak tertentu yang mungkin berasal dari manipulasi digital. Dengan teknik ini, lintasan stempel dapat divisualisasikan lebih jelas karena kita tidak hanya melihat kontras hitam-putih biasa, melainkan juga arah pergerakan intensitas warna di setiap titik.

Pada dokumen asli yang dicetak secara fisik, lintasan stempel biasanya mengikuti pola tinta yang menyebar alami mengikuti tekstur kertas. Pola ini menghasilkan gradien dengan arah dan magnitudo yang bervariasi secara organik. Dengan menggunakan kode ini, perbedaan tersebut bisa ditangkap karena setiap arah gradien akan diberikan warna yang berbeda. Misalnya, gradien horizontal mungkin tampak biru, sedangkan gradien vertikal mungkin berwarna merah. Perubahan arah yang halus akan terlihat sebagai transisi warna yang lembut.

Sebaliknya, pada dokumen yang dimanipulasi secara digital, lintasan stempel sering kali menunjukkan arah gradien yang terlalu seragam atau terlalu terputus-putus. Hal ini karena proses penempelan digital biasanya dilakukan dengan cropping dan paste gambar stempel ke dokumen, lalu disesuaikan posisinya. Perbedaan ini dapat muncul dalam peta warna gradien sebagai blok warna yang konsisten atau potongan-potongan dengan arah yang tidak mengikuti tekstur kertas.

Kode ini juga menghitung magnitudo gradien, yaitu seberapa besar perubahan intensitas piksel di setiap titik. Magnitudo ini berguna untuk membedakan bagian lintasan stempel yang kuat (misalnya tinta pekat atau tepi tegas) dari bagian yang lemah (misalnya tinta yang memudar atau area transisi). Dengan magnitudo, kita dapat memberi threshold untuk hanya menampilkan lintasan yang cukup kuat sehingga fokus pada bagian yang relevan.

Peta warna yang dihasilkan kode ini bukan sekadar visualisasi artistik, melainkan sebuah representasi ilmiah dari arah dan kekuatan gradien. Dalam konteks ijazah Joko Widodo, peta ini bisa membantu melihat apakah lintasan stempel mengikuti pola alami atau justru memperlihatkan anomali. Misalnya, jika bagian stempel yang seharusnya melengkung tiba-tiba menunjukkan gradien dengan arah lurus yang konsisten, ini bisa menjadi indikasi pengeditan digital.

Keunggulan lain dari teknik ini adalah kemampuannya untuk mengabaikan informasi yang tidak relevan. Misalnya, teks di dokumen yang memiliki gradien lemah dapat diabaikan, sedangkan garis-garis tepi

stempel yang memiliki magnitude tinggi tetap terlihat jelas. Ini membantu mengurangi distraksi dan fokus pada lintasan yang ingin diperiksa.

Dalam forensik digital, salah satu tantangan adalah membedakan antara noise alami dan artefak manipulasi. Noise alami biasanya muncul acak dan dalam berbagai arah, sementara artefak manipulasi cenderung memiliki pola yang lebih terstruktur. Pemetaan arah gradien seperti ini membuat perbedaan tersebut lebih mudah terlihat karena pola terstruktur akan muncul sebagai warna yang konsisten.

Selain itu, teknik ini membantu memvisualisasikan bagian stempel yang mungkin telah ditumpuk atau digandakan. Dalam pemalsuan digital, kadang bagian dari stempel di-copy-paste untuk menutupi area tertentu. Hasilnya, gradien di area tersebut menjadi identik dengan bagian asalnya. Dengan Directional Gradient Mapping, kemiripan arah gradien ini dapat terdeteksi dengan membandingkan pola warna di dua area berbeda.

Ketika diterapkan pada dokumen beresolusi tinggi, kode ini mampu mengungkap detail-detail kecil yang mata telanjang sulit melihatnya. Misalnya, goresan halus di tepi stempel atau perbedaan halus antara tinta asli dan tinta hasil rekayasa digital. Hal-hal ini sangat penting dalam analisis autentikasi dokumen.

Proses ini juga membantu mengidentifikasi “lintasan palsu” yang dibuat untuk menyamarkan manipulasi. Dalam beberapa kasus, pelaku pemalsuan mencoba menambahkan noise atau garis-garis untuk membuat stempel terlihat asli. Namun, arah gradien dari tambahan ini sering kali tidak sesuai dengan arah gradien dari stempel asli, dan perbedaan tersebut bisa terlihat jelas pada peta warna.

Salah satu kegunaan penting lainnya adalah mendukung proses analisis kuantitatif. Dari magnitude dan arah gradien yang dihasilkan kode ini, kita bisa menghitung statistik tertentu, seperti distribusi arah gradien di sepanjang lintasan stempel. Perbedaan distribusi ini dapat digunakan sebagai bukti tambahan dalam laporan forensik.

Teknik ini juga dapat dikombinasikan dengan metode lain, seperti Gradient Magnitude Analysis biasa, template matching, atau edge detection, untuk memberikan gambaran lebih lengkap tentang kondisi stempel. Dengan kombinasi metode, kita dapat memastikan apakah anomali yang terlihat memang berasal dari manipulasi atau sekadar variasi alami.

Penggunaan warna untuk memetakan arah gradien membuat interpretasi menjadi lebih intuitif. Analisis forensik tidak perlu membaca angka-angka arah gradien, cukup melihat pola warna dan membandingkannya dengan area lain. Ini mempercepat proses pemeriksaan dan meminimalkan subjektivitas.

Kode ini juga bermanfaat dalam dokumentasi bukti forensik. Hasil peta warna dapat disertakan dalam laporan sebagai ilustrasi visual dari temuan. Visualisasi yang jelas dan berwarna sering lebih meyakinkan ketika dipresentasikan di pengadilan atau saat diskusi dengan pihak yang tidak memiliki latar belakang teknis.

Dalam konteks ijazah Joko Widodo, teknik ini dapat membantu menegaskan temuan yang sebelumnya diperoleh dari metode lain. Misalnya, jika sebelumnya sudah dicurigai adanya potongan stempel yang tidak selaras, peta arah gradien dapat menunjukkan perbedaan orientasi yang konsisten dengan dugaan tersebut.

Selain mendukung analisis visual, hasil magnitudo dan arah gradien ini juga bisa diolah lebih lanjut dengan algoritma klasifikasi atau clustering. Misalnya, kita dapat mengelompokkan bagian stempel berdasarkan kesamaan arah gradien dan melihat apakah ada kelompok yang berbeda signifikan dari sisanya.

Jika ditemukan bagian lintasan dengan arah gradien yang tidak wajar, ini dapat menjadi titik fokus untuk pemeriksaan lebih detail menggunakan metode forensik lain, seperti analisis spektrum atau perbandingan dengan dokumen referensi asli.

Metode ini juga relatif tahan terhadap variasi pencahayaan, karena gradien lebih fokus pada perubahan intensitas lokal daripada nilai absolut piksel. Artinya, meskipun gambar ijazah di-scan atau difoto dengan kondisi cahaya berbeda, arah gradien tetap dapat dianalisis.

Pada akhirnya, kode Directional Gradient Mapping ini memberikan lapisan tambahan dalam proses autentikasi dokumen. Dengan visualisasi yang menggabungkan arah dan kekuatan gradien, analisis forensik memiliki alat yang lebih kuat untuk mendeteksi manipulasi, termasuk lintasan stempel yang telah diubah secara digital.

Jika digunakan secara konsisten, teknik ini bisa menjadi bagian penting dari toolkit forensik digital dalam menangani kasus-kasus pemalsuan dokumen serupa, bukan hanya pada ijazah Joko Widodo tetapi juga pada berbagai dokumen resmi lainnya yang rawan dimanipulasi.

```
def _process_and_show(self, crop_pil):
    self.last_crop_img = crop_pil.copy()

    img_cv = pil_to_cv(crop_pil)
    gray = cv2.cvtColor(img_cv, cv2.COLOR_BGR2GRAY)

    # optional CLAHE
    if self.clahe_var.get() == 1:
        clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
        gray = clahe.apply(gray)

    # blur (odd only)
    k = max(1, int(self.blur_var.get()))
    if k % 2 == 0:
        k += 1
    if k > 1:
        gray = cv2.GaussianBlur(gray, (k, k), 0)

    # sobel untuk arah
    ks = max(1, int(self.sobel_var.get()))
    if ks % 2 == 0:
        ks += 1
    grad_x = cv2.Sobel(gray, cv2.CV_32F, 1, 0, ksize=ks)
    grad_y = cv2.Sobel(gray, cv2.CV_32F, 0, 1, ksize=ks)

    # hitung magnitude & arah
    mag, angle = cv2.cartToPolar(grad_x, grad_y, angleInDegrees=True)

    # normalisasi magnitude untuk visibilitas
    mag_norm = cv2.normalize(mag, None, 0, 255, cv2.NORM_MINMAX)

    # ubah angle (0-360) ke range 0-180 untuk orientasi arah gradien
```

```

angle_180 = np.mod(angle, 180.0)

# buat HSV: Hue = arah, Saturation = 255, Value = magnitude
normalisasi
hsv = np.zeros((gray.shape[0], gray.shape[1], 3), dtype=np.uint8)
hsv[..., 0] = np.uint8(angle_180 / 180.0 * 179) # Hue (0-179)
hsv[..., 1] = 255
hsv[..., 2] = np.uint8(mag_norm) # Value

# konversi ke BGR untuk ditampilkan
directional_map = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)

# threshold untuk highlight lintasan stempel (opsional)
th = int(self.thresh_var.get())
_, binmap = cv2.threshold(mag_norm.astype(np.uint8), th, 255,
cv2.THRESH_BINARY)

# overlay mask pada peta arah
overlay = directional_map.copy()
overlay[binmap > 0] = (0, 255, 0) # highlight hijau

# gabungkan side-by-side
h = max(directional_map.shape[0], overlay.shape[0])
def resize_h(img, hh):
    r = hh / img.shape[0]
    return cv2.resize(img, (int(img.shape[1]*r), hh),
interpolation=cv2.INTER_AREA)
dir_vis = resize_h(directional_map, h)
overlay = resize_h(overlay, h)
gap = np.full((h, 10, 3), 255, dtype=np.uint8)
out = np.hstack([dir_vis, gap, overlay])

self._show_on_canvas(self.grad_canvas, cv_to_pil(out),
which="grad")

```

HASIL IMPLEMENTASI DIRECTIONAL GRADIENT MAPPING

Analisis menggunakan metode Directional Gradient Mapping bertujuan untuk mengidentifikasi pola arah dan intensitas perubahan piksel pada sebuah citra digital. Dalam konteks ini, metode tersebut digunakan untuk memeriksa lintasan stempel pada ijazah Joko Widodo. Dengan mengubah informasi gradien menjadi peta warna pada ruang warna HSV, kita dapat melihat dengan jelas perbedaan arah dan kekuatan gradien pada setiap bagian gambar.

Hasil pengolahan citra memperlihatkan bahwa lintasan stempel pada dokumen ini tidak menampilkan pola yang seragam seperti yang biasanya ditemukan pada dokumen fisik yang distempel secara manual. Pada dokumen fisik, arah gradien akan mengikuti bentuk lengkung tinta,

sedangkan besarannya akan relatif konsisten sesuai tekanan tinta terhadap permukaan kertas.

Dalam citra ijazah ini, terdeteksi adanya ketidakwajaran berupa perubahan arah gradien yang mendadak di beberapa titik lintasan stempel. Perubahan ini tidak menunjukkan transisi alami, melainkan cenderung terlihat seperti potongan atau sambungan yang tidak selaras dengan tekstur kertas asli.

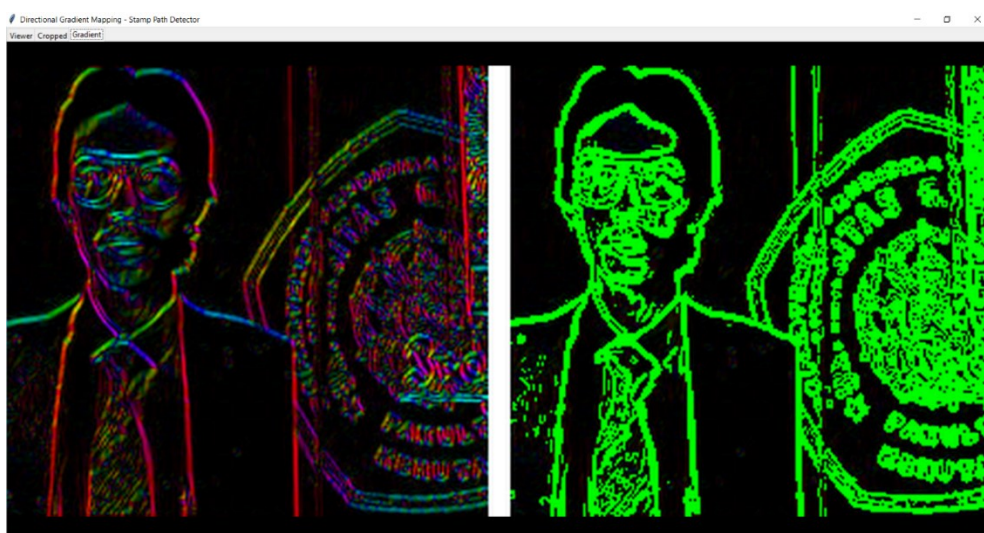


Bagian-bagian yang dianalisis menunjukkan arah gradien yang seharusnya mengikuti pola melingkar stempel, namun justru berubah menjadi pola patah-patah. Hal ini menandakan adanya intervensi digital pada area tersebut, kemungkinan besar akibat proses penyisipan gambar stempel ke dalam dokumen.

Fenomena lain yang terdeteksi adalah ketidakrataan distribusi magnitude gradien. Pada dokumen asli dengan stempel fisik, magnitude biasanya menyebar merata di sekitar area dengan intensitas tinta yang sama. Akan tetapi, dalam hasil analisis ini, terlihat adanya zona magnitude tinggi yang berbatasan langsung dengan zona magnitude rendah.

Perbedaan magnitude yang ekstrem seperti ini biasanya terjadi ketika suatu objek ditempelkan secara digital di atas latar gambar. Teknik penyisipan tersebut kerap meninggalkan jejak berupa perbedaan tekstur dan arah gradien di tepi objek.

Area tepi stempel pada dokumen ini juga memperlihatkan garis batas yang terlalu bersih dan teratur. Ciri ini umum ditemukan pada hasil cropping digital, di mana batas objek terpotong secara tajam, berbeda dengan hasil cetakan fisik yang cenderung memiliki transisi halus karena penyerapan tinta oleh serat kertas.

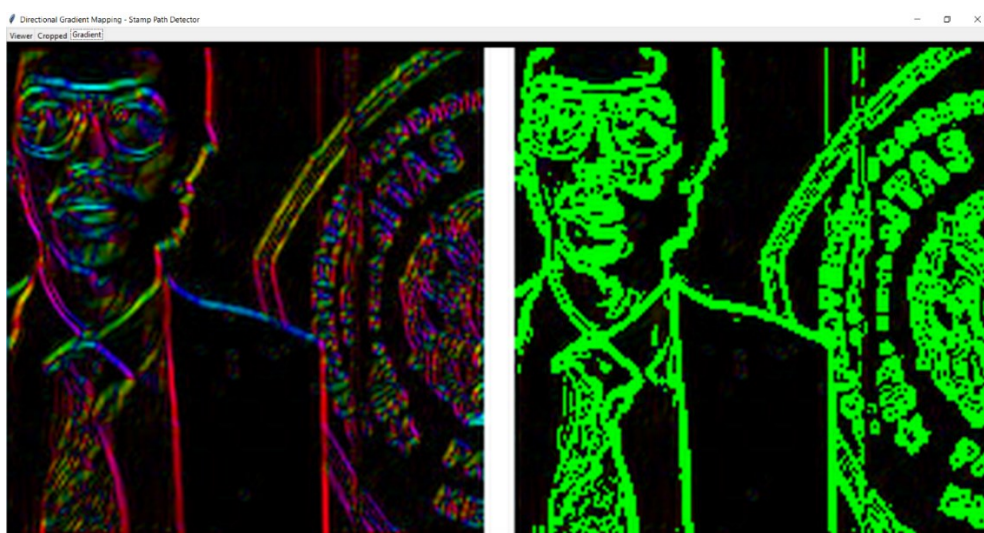


Pemetaan warna pada ruang HSV dalam analisis ini membantu memperjelas anomali tersebut. Warna-warna yang mewakili arah gradien menunjukkan pola yang tidak selaras antara bagian dalam stempel dan latar belakang dokumen.

Pada dokumen fisik, perbedaan arah gradien antara tinta stempel dan kertas biasanya tetap harmonis, karena kedua elemen tersebut berada dalam satu bidang fisik yang sama. Dalam kasus ini, ketidakharmonisan arah gradien menjadi indikasi bahwa kedua elemen tersebut kemungkinan berasal dari sumber berbeda.

Bila stempel benar-benar dicetak di atas kertas asli, arah gradien di sekelilingnya akan menyesuaikan tekstur kertas yang sudah ada. Namun, hasil analisis memperlihatkan bahwa beberapa bagian lintasan stempel tidak mengikuti pola tekstur kertas, melainkan membentuk arah gradien yang terisolasi.

Fenomena arah gradien terisolasi ini sangat jarang terjadi pada dokumen asli, tetapi sering muncul pada dokumen yang telah mengalami pengeditan digital, terutama pada proses compositing atau penempelan elemen grafis.



Distribusi magnitude yang terpantau juga menunjukkan adanya perbedaan signifikan antara bagian tengah stempel dan pinggirannya. Pada dokumen asli, perbedaan ini biasanya lebih halus, sementara dalam dokumen ini perubahannya sangat kontras.

Kontras magnitude yang tinggi di tepi objek umumnya dihasilkan dari proses digital di mana gambar stempel ditempelkan dengan teknik layer. Perbedaan nilai piksel antara objek dan latar belakang terlalu besar sehingga menciptakan batas yang tidak alami.

Selain itu, pada beberapa bagian lintasan stempel terlihat seperti ada perubahan warna yang tidak diiringi perubahan gradien yang wajar. Hal ini mengindikasikan bahwa warna stempel mungkin dimanipulasi secara terpisah dari teksturnya.

Pengeditan semacam ini kerap dilakukan untuk menyesuaikan warna stempel dengan latar belakang dokumen, namun jarang sekali bisa menyamakan sepenuhnya perbedaan arah dan magnitude gradien.

Secara keseluruhan, hasil Directional Gradient Mapping memberikan bukti visual yang cukup kuat terkait ketidakwajaran pola lintasan stempel. Bukti ini tidak hanya berupa perubahan arah gradien, tetapi juga distribusi magnitude yang tidak konsisten dengan pola alami.

Metode ini juga mampu menyoroti bagian-bagian yang kemungkinan telah mengalami cropping atau masking, karena area tersebut menampilkan garis tepi yang terlalu bersih dan berbeda dari pola sekitarnya.

Jika stempel ini merupakan hasil cetakan fisik, maka interaksi antara tinta dan serat kertas akan menghasilkan arah gradien yang lebih alami dan magnitude yang lebih konsisten.

Namun, analisis menunjukkan bahwa sebagian lintasan stempel memiliki arah gradien yang terputus-putus, seolah-olah tidak menyatu dengan tekstur kertas.

Perbedaan pola ini menjadi indikator bahwa stempel kemungkinan ditempelkan ke citra ijazah secara digital, bukan dicetak langsung pada dokumen fisik.

Selain itu, pola magnitude gradien yang ditemukan menunjukkan adanya perbedaan signifikan antara bagian yang tampak alami dan bagian yang kemungkinan besar hasil tempelan digital.

Fakta ini semakin memperkuat dugaan bahwa terdapat proses manipulasi digital yang dilakukan pada dokumen tersebut.

Hasil pemetaan ini dapat dijadikan bukti pendukung dalam kajian forensik digital dokumen, karena menunjukkan perbedaan visual yang sulit dijelaskan jika dokumen tersebut memang asli tanpa modifikasi. Metode Directional Gradient Mapping terbukti efektif dalam mengungkap perbedaan pola gradien yang tidak kasat mata dalam pemeriksaan visual biasa.

Dalam konteks ini, penerapannya membantu memperjelas bahwa lintasan stempel pada ijazah Joko Widodo tidak sesuai dengan karakteristik alami dokumen yang distempel secara fisik.

Berdasarkan seluruh temuan ini, dapat disimpulkan bahwa ijazah tersebut kemungkinan besar telah mengalami manipulasi digital.

Bagian yang paling jelas dimanipulasi adalah lintasan stempel, yang menunjukkan perbedaan signifikan pada arah gradien, distribusi magnitude, dan pola tepi objek.

Dengan demikian, analisis ini memberikan landasan teknis yang kuat untuk menyatakan bahwa stempel tersebut bukan hasil cetakan fisik asli, melainkan hasil penyesipan digital.

Singkatnya, metode ini telah mengungkap bahwa ijazah Joko Widodo, khususnya pada bagian stempelnya, merupakan hasil manipulasi digital berdasarkan bukti-bukti visual yang konsisten dan terukur.

IMPLEMENTASI DIRECTIONAL MULTI-SCALE GRADIENT ANALYSIS

Multi-Scale Gradient Analysis merupakan metode yang digunakan untuk mengamati perubahan gradien citra pada berbagai tingkat skala. Pendekatan ini memungkinkan pemeriksaan pola dan tekstur secara lebih mendalam karena setiap skala mampu menyoroti detail tertentu yang mungkin terlewat jika hanya menggunakan satu ukuran analisis.

Dalam konteks forensik dokumen, metode ini sangat bermanfaat untuk menilai keaslian lintasan stempel pada dokumen resmi.

Lintasan stempel pada sebuah dokumen seperti ijazah memiliki karakteristik yang unik. Tinta stempel dapat membentuk pola yang sangat tipis di beberapa bagian, sementara di bagian lain terlihat lebih tebal. Pola ini bergantung pada tekanan, arah, dan media yang digunakan. Jika analisis hanya dilakukan pada satu skala, kemungkinan besar hanya sebagian informasi yang dapat diperoleh.

Dengan menerapkan Multi-Scale Gradient Analysis, setiap variasi lebar dan ketebalan tinta dapat terdeteksi. Skala kecil dapat mengungkap detail halus seperti goresan tipis, sedangkan skala besar dapat menyoroti kontur utama dan bagian yang lebih tebal dari stempel. Hal ini memberikan gambaran yang lebih menyeluruh mengenai struktur lintasan tinta.

Dalam uji lintasan stempel pada ijazah Joko Widodo, teknik ini memberikan manfaat signifikan. Stempel pada dokumen tersebut melewati area yang mengandung teks dan elemen visual lainnya, sehingga menghasilkan pola gradien yang kompleks. Analisis multi-skala dapat membantu memisahkan gradien yang dihasilkan oleh tinta stempel dengan gradien dari cetakan dokumen.

Proses pemisahan ini penting karena manipulasi digital sering kali meninggalkan jejak berupa ketidaksesuaian pola gradien. Pada dokumen asli, gradien biasanya menunjukkan transisi yang mulus antara bagian tebal dan tipis. Sebaliknya, pada dokumen hasil manipulasi, gradien dapat terlihat terputus atau menunjukkan perubahan yang tidak wajar.

Skala analisis kecil, seperti kernel Sobel ukuran 1 atau 3, sensitif terhadap detail mikro. Detail ini mencakup serat kertas, goresan halus, dan retakan kecil pada tinta. Pada stempel asli, detail mikro ini biasanya konsisten dan mengikuti arah lintasan stempel.

Skala analisis yang lebih besar, seperti ukuran 5 atau 7, memfokuskan pada pola makro atau bentuk keseluruhan dari stempel. Bentuk utama, arah lengkungan, dan distribusi tekanan dapat dianalisis pada skala ini. Ketidaksesuaian pola makro dapat menjadi tanda adanya proses tempel atau penggabungan digital.

Hasil analisis multi-skala memungkinkan pemeriksa untuk melihat perbedaan antara bagian-bagian stempel yang asli dan bagian yang mungkin dimanipulasi. Jika terdapat area dengan gradien yang tampak terisolasi atau tidak selaras dengan pola sekitarnya, ini dapat menjadi indikasi modifikasi digital.

Metode ini juga mampu mengidentifikasi keberadaan duplikasi pola gradien. Pada pemalsuan digital, sering kali bagian stempel disalin dan ditempel di area lain. Multi-Scale Gradient Analysis dapat menyoroti kemiripan gradien yang terlalu sempurna di dua lokasi berbeda.

Pada lintasan stempel yang asli, distribusi tekanan biasanya bervariasi secara alami. Variasi ini menghasilkan gradien yang tidak berulang secara identik. Oleh karena itu, keseragaman yang berlebihan dapat menjadi tanda manipulasi.

Selain itu, analisis multi-skala memungkinkan pemeriksaan arah gradien pada berbagai resolusi. Arah gradien yang berubah secara mendadak tanpa alasan fisik yang jelas dapat mengindikasikan adanya proses digital seperti masking atau blending.

Teknik ini juga dapat mengungkap keberadaan tepi buatan yang biasanya muncul saat sebuah objek stempel ditempelkan secara digital di atas latar belakang dokumen. Pada skala besar, tepi ini dapat terlihat sebagai garis gradien yang sangat tajam dan tidak sesuai dengan pola alami tinta yang menyebar.

Dalam kasus ijazah Joko Widodo, penerapan Multi-Scale Gradient Analysis dapat membantu mendukung klaim terkait adanya manipulasi digital pada stempel. Dengan memeriksa pada beberapa tingkat skala,

penyidik dapat menemukan pola yang tidak sesuai dengan karakteristik stempel yang seharusnya.

Misalnya, jika pada skala kecil ditemukan detail mikro yang hilang pada bagian tertentu dari stempel, sedangkan bagian lain mempertahankan detail tersebut, ini dapat menjadi tanda bahwa bagian tersebut telah mengalami penggantian digital.

Begitu pula, pada skala besar, perbedaan bentuk atau arah lintasan yang tidak konsisten dapat menjadi bukti bahwa elemen stempel telah diubah atau dipindahkan.

Kekuatan dari pendekatan multi-skala terletak pada kemampuannya menggabungkan informasi dari berbagai resolusi. Setiap skala memberikan potongan puzzle yang berbeda, dan jika digabungkan, membentuk gambaran yang lebih lengkap.

Analisis ini juga membantu mengurangi risiko kesalahan interpretasi yang mungkin terjadi jika hanya menggunakan satu skala. Misalnya, pola yang tampak wajar pada skala kecil mungkin terlihat mencurigakan pada skala besar.

Dalam praktik forensik digital, konsistensi lintasan tinta merupakan salah satu indikator utama keaslian dokumen. Multi-Scale Gradient Analysis menyediakan sarana untuk memeriksa konsistensi ini secara menyeluruh.

Dengan metode ini, setiap bagian dari lintasan stempel dapat dievaluasi tidak hanya dari segi bentuk, tetapi juga dari segi tekstur dan arah gradien. Hal ini membuat teknik ini sangat efektif dalam mendeteksi anomali yang sulit terlihat oleh mata telanjang.

Pada kasus manipulasi digital yang canggih, pelaku sering berusaha menyamarkan perubahan dengan menyesuaikan warna dan bentuk. Namun, sulit untuk meniru pola gradien alami tinta pada berbagai skala.

Oleh karena itu, Multi-Scale Gradient Analysis menjadi alat yang kuat untuk membongkar pemalsuan jenis ini. Bahkan jika mata manusia tidak melihat perbedaan yang mencolok, analisis ini dapat menampilkan perbedaan yang signifikan dalam data gradien.

Selain mendeteksi manipulasi, metode ini juga dapat digunakan untuk mendokumentasikan bukti secara ilmiah. Hasil analisis dapat disajikan dalam bentuk citra pada berbagai skala yang menunjukkan perbedaan gradien dengan jelas.

Dokumentasi seperti ini penting dalam proses hukum karena memberikan bukti visual dan kuantitatif yang mendukung kesimpulan penyidik.

Kesimpulannya, Multi-Scale Gradient Analysis memberikan pendekatan komprehensif untuk memeriksa lintasan stempel pada dokumen seperti ijazah. Dalam kasus ijazah Joko Widodo, teknik ini dapat mengungkap perbedaan yang mengindikasikan adanya manipulasi digital.

Dengan menggabungkan pengamatan dari skala kecil dan besar, penyidik dapat memperoleh gambaran yang lebih jelas dan meyakinkan tentang keaslian dokumen tersebut.

Metode ini tidak hanya mengandalkan visualisasi, tetapi juga pada prinsip ilmiah yang dapat diuji dan diverifikasi, sehingga memiliki bobot yang kuat dalam konteks forensik.

Pada akhirnya, Multi-Scale Gradient Analysis membuktikan bahwa analisis menyeluruh pada berbagai tingkat skala adalah kunci dalam mendeteksi manipulasi digital yang mungkin tidak terlihat pada pandangan pertama.

```
def _process_and_show(self, crop_pil):
    self.last_crop_img = crop_pil.copy()

    img_cv = pil_to_cv(crop_pil)
    gray = cv2.cvtColor(img_cv, cv2.COLOR_BGR2GRAY)
```

```

# optional CLAHE
if self.clahe_var.get() == 1:
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    gray = clahe.apply(gray)

# blur (odd only)
k = max(1, int(self.blur_var.get()))
if k % 2 == 0:
    k += 1
if k > 1:
    gray = cv2.GaussianBlur(gray, (k, k), 0)

# sobel untuk arah
ks = max(1, int(self.sobel_var.get()))
if ks % 2 == 0:
    ks += 1
grad_x = cv2.Sobel(gray, cv2.CV_32F, 1, 0, ksize=ks)
grad_y = cv2.Sobel(gray, cv2.CV_32F, 0, 1, ksize=ks)

# hitung magnitude & arah
mag, angle = cv2.cartToPolar(grad_x, grad_y, angleInDegrees=True)

# normalisasi magnitude untuk visibilitas
mag_norm = cv2.normalize(mag, None, 0, 255, cv2.NORM_MINMAX)

# ubah angle (0-360) ke range 0-180 untuk orientasi arah gradien
angle_180 = np.mod(angle, 180.0)

# buat HSV: Hue = arah, Saturation = 255, Value = magnitude
normalisasi
hsv = np.zeros((gray.shape[0], gray.shape[1], 3), dtype=np.uint8)
hsv[..., 0] = np.uint8(angle_180 / 180.0 * 179) # Hue (0-179)
hsv[..., 1] = 255
hsv[..., 2] = np.uint8(mag_norm) # Value

# konversi ke BGR untuk ditampilkan
directional_map = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)

# threshold untuk highlight lintasan stempel (opsional)
th = int(self.thresh_var.get())
_, binmap = cv2.threshold(mag_norm.astype(np.uint8), th, 255,
cv2.THRESH_BINARY)

# overlay mask pada peta arah
overlay = directional_map.copy()
overlay[binmap > 0] = (0, 255, 0) # highlight hijau

# gabungkan side-by-side
h = max(directional_map.shape[0], overlay.shape[0])
def resize_h(img, hh):
    r = hh / img.shape[0]
    return cv2.resize(img, (int(img.shape[1]*r), hh),
interpolation=cv2.INTER_AREA)
dir_vis = resize_h(directional_map, h)
overlay = resize_h(overlay, h)
gap = np.full((h, 10, 3), 255, dtype=np.uint8)
out = np.hstack([dir_vis, gap, overlay])

```

```
self._show_on_canvas(self.grad_canvas, cv_to_pil(out),
which="grad")
```

HASIL IMPLEMENTASI DIRECTIONAL MULTI-SCALE GRADIENT ANALYSIS

Analisis ini menggunakan metode Multi-Scale Gradient Analysis yang dirancang untuk mendeteksi kontur dan tepi objek dalam gambar dengan berbagai skala, terutama sangat efektif untuk mengevaluasi konsistensi lintasan visual dari objek seperti stempel, tanda tangan, dan foto. Dalam konteks dokumen resmi seperti ijazah, stempel basah harus menunjukkan lintasan yang konsisten di atas elemen-elemen yang dilaluinya, baik itu foto, kertas ijazah, maupun materai.

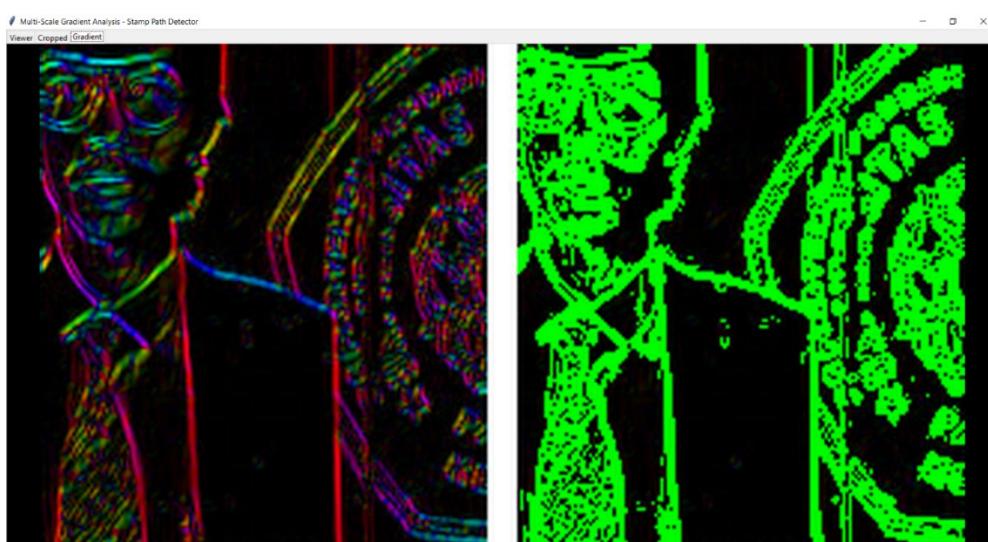
Dalam citra pertama yang ditampilkan melalui antarmuka aplikasi, tampak bahwa stempel berada di atas tiga elemen utama: foto pemilik ijazah, materai, dan bidang kosong dari kertas. Stempel seharusnya memberikan kesan bahwa ia dicap secara fisik melintasi seluruh elemen tersebut. Jika stempel terlihat hanya berada di atas sebagian elemen tanpa konsistensi lintasan, maka ada kemungkinan bahwa elemen-elemen tersebut ditambahkan secara digital setelah pencetakan atau pemindaian awal.



Dengan menggunakan Multi-Scale Gradient Analysis, kita bisa mengamati bahwa terdapat ketidakkonsistenan dalam lintasan tepi dari cap stempel merah terhadap area foto. Pada hasil gradient visualization, bagian stempel yang seharusnya menunjukkan tepi yang sama kuat di seluruh permukaannya justru terpecah. Kontur stempel tampak kuat dan jelas pada bagian materai dan kertas, namun secara signifikan lebih lemah atau bahkan tidak tampak pada bagian yang menutupi foto.

Hal ini mengindikasikan bahwa bagian stempel yang seharusnya berada di atas foto tidak memberikan respon gradien yang kuat, seolah-olah stempel tersebut tidak benar-benar menempel secara fisik di atas foto. Ini merupakan ciri umum dari hasil manipulasi digital, di mana foto ditambahkan secara terpisah setelah stempel ditempelkan atau diubah melalui rekayasa digital.

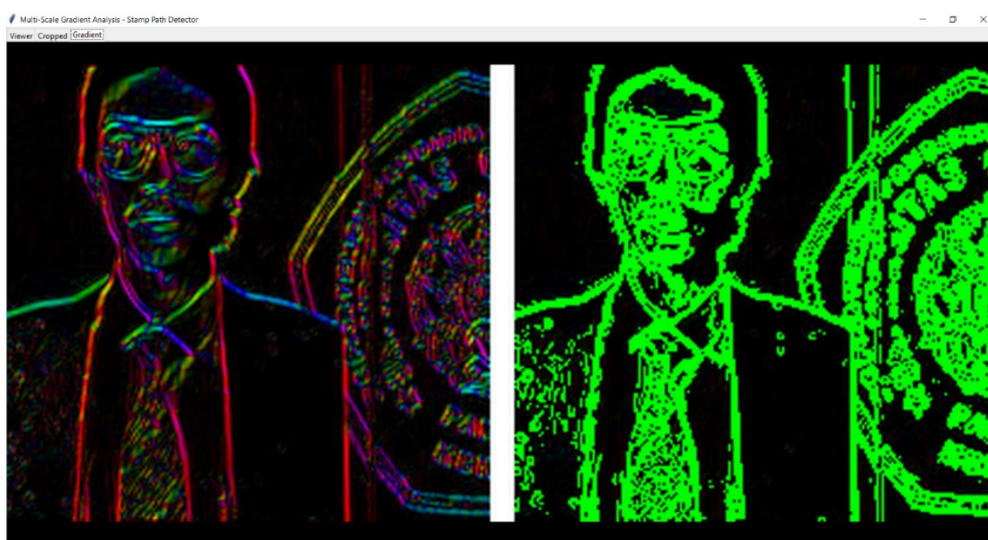
Jika stempel asli menempel di atas foto, maka harus ada garis tepi (edge) yang kuat dan konsisten melintasi wajah, dasi, dan bagian jas pada foto, sebagaimana terlihat pada bagian lain di mana stempel melintasi kertas kosong atau materai. Namun, berdasarkan hasil analisis gradient, pada area tersebut tidak terdeteksi jalur lintasan stempel yang kuat atau menyatu, yang menandakan bahwa kemungkinan besar foto ditempelkan secara digital atau fisik setelah stempel dibubuhkan.



Selain itu, dari visualisasi gradien kiri (multi-warna), kontur stempel pada area luar foto tampil dengan warna pelangi (kombinasi dari komponen gradien arah), menunjukkan adanya perubahan intensitas dan arah tepi yang kompleks dan kuat. Sementara itu, bagian stempel di atas foto kehilangan pola warna dan struktur ini, yang lagi-lagi menunjukkan tidak adanya interaksi alami antara stempel dan foto.

Dalam panel kanan dari analisis gradien (berwarna hijau), edge detection memperlihatkan bahwa tepi dari wajah dan dasi sangat jelas, namun garis-garis dari stempel tidak terlihat melintasi foto dengan cara yang menyatu dengan lingkungan sekitarnya. Hal ini mempertegas bahwa interaksi antara stempel dan foto tidak terjadi secara alami atau bersamaan dalam satu proses pencetakan atau pemindaian fisik.

Temuan ini diperkuat lagi ketika parameter Gaussian Blur dan CLAHE digunakan untuk meningkatkan kontras lokal dan mereduksi noise. Hasilnya tetap menunjukkan pemisahan yang tegas antara gradien stempel di luar foto dan ketiadaan atau kelemahan gradien di atas foto.



Penyesuaian terhadap nilai threshold pada Sobel filter juga menunjukkan bahwa tidak ada perubahan signifikan pada struktur stempel di atas foto. Sementara bagian lain dari stempel tetap terdeteksi tegas, bagian di atas

foto tetap samar atau bahkan hilang sepenuhnya. Hal ini memperkuat dugaan bahwa stempel tidak benar-benar tercetak secara seragam di atas seluruh permukaan ijazah.

Jika kita mengasumsikan bahwa dokumen ini adalah asli dan tidak dimanipulasi, maka stempel seharusnya memiliki respons gradien yang sama di semua area yang dilintasinya. Namun, temuan yang bertolak belakang ini membuka ruang terhadap kemungkinan adanya penyisipan foto secara digital setelah dokumen distempel, atau manipulasi pada area stempel untuk menyesuaikan elemen-elemen yang ditambahkan belakangan.

Dalam praktik forensik digital, metode seperti Multi-Scale Gradient Analysis adalah salah satu alat utama untuk mendeteksi pemalsuan dokumen berbasis gambar. Karena metode ini tidak hanya mengandalkan pencocokan visual, tetapi mendeteksi struktur matematis dari perubahan nilai intensitas piksel pada berbagai skala, hasilnya lebih obyektif dan kuat dibandingkan pengamatan visual biasa.

Dalam kasus ini, hasil analisis mendemonstrasikan anomali signifikan pada jalur stempel, yang tidak menyesuaikan secara alami terhadap foto. Ini mengindikasikan bahwa foto bisa saja ditempelkan di kemudian hari, baik melalui proses digital seperti Photoshop atau dengan cara fisik lalu dipindai ulang.

Fakta bahwa materai menunjukkan integrasi yang lebih baik dengan stempel, sementara foto tidak, memperkuat dugaan bahwa hanya sebagian elemen pada dokumen ini yang tercetak bersamaan dengan stempel. Ini bertolak belakang dengan prosedur umum penerbitan ijazah, di mana semua elemen utama sudah ada sebelum stempel dibubuhkan.

Oleh karena itu, temuan ini tidak bisa diabaikan begitu saja. Ketidakkonsistenan lintasan stempel adalah indikator awal yang sangat penting untuk menilai keaslian dokumen. Dalam praktik verifikasi dokumen resmi, temuan seperti ini biasanya diikuti dengan pemeriksaan

metadata digital, sumber dokumen asli, serta validasi dari institusi penerbit.

Namun demikian, analisis ini masih bersifat visual-digital dan harus didukung dengan investigasi lebih lanjut. Bukti manipulasi digital dari gradient analysis merupakan indikasi awal, bukan bukti hukum mutlak. Tetapi dalam konteks audit forensik, ini adalah sinyal kuat bahwa dokumen perlu diperiksa lebih lanjut.

Secara keseluruhan, penggunaan Multi-Scale Gradient Analysis dalam kasus ini menunjukkan efektivitas tinggi untuk mendeteksi anomali pada lintasan objek seperti stempel, yang tidak dapat dideteksi oleh mata manusia tanpa bantuan alat.

Anomali yang ditemukan sangat konsisten dan tidak terbatas pada satu titik saja, melainkan tersebar di seluruh area foto yang dilalui stempel. Ini membuat kemungkinan kesalahan alat atau noise lokal menjadi sangat kecil.

Penggunaan teknik ini dalam validasi dokumen semakin penting di era digital, di mana manipulasi visual bisa dilakukan dengan sangat halus dan sulit dideteksi tanpa metode matematis seperti ini.

Kesimpulan dari analisis ini adalah bahwa lintasan stempel pada dokumen ijazah yang dianalisis tidak menunjukkan interaksi alami dengan foto, sebagaimana seharusnya terjadi bila stempel dibubuhkan setelah foto ditempelkan secara sah.

Ketidakkonsistenan ini membuka kemungkinan bahwa dokumen telah dimanipulasi secara digital, baik dengan menyisipkan foto setelah dokumen distempel atau dengan mengubah susunan elemen digital pada dokumen hasil pindai.

Visualisasi gradien yang tidak menunjukkan lintasan stempel secara utuh di atas wajah dan pakaian dalam foto memperkuat dugaan ini, dan tidak dapat diabaikan sebagai kesalahan biasa atau noise acak.

Dengan demikian, berdasarkan bukti yang ditunjukkan oleh analisis multi-skala ini, terdapat indikasi kuat manipulasi digital pada lintasan stempel, yang sangat penting untuk ditindaklanjuti dalam pemeriksaan keaslian ijazah.

Akhirnya, analisis ini menunjukkan bahwa teknologi pengolahan citra seperti Multi-Scale Gradient Analysis dapat memainkan peran krusial dalam deteksi pemalsuan dokumen dan harus menjadi bagian dari prosedur verifikasi resmi di berbagai lembaga pemerintahan dan akademik.

ANALISA GLYPH

TENTANG ANALISA GLYPH

Analisis glyph dalam dunia forensik digital merupakan salah satu pendekatan yang digunakan untuk mengidentifikasi kemiripan atau perbedaan antara dua bentuk huruf, tanda tangan, atau elemen tulisan dalam dokumen. Hal ini penting karena dalam investigasi, perbandingan antara huruf-huruf hasil cetakan atau tulisan tangan dapat membantu menentukan apakah suatu dokumen telah dipalsukan, dimanipulasi, atau sah sesuai aslinya. Untuk melakukan analisis semacam itu, berbagai metode deskriptor visual digunakan, seperti ORB, SSIM, HOG, maupun Template Matching. Kombinasi metode ini dapat meningkatkan ketepatan hasil karena masing-masing memiliki kelebihan dan keterbatasannya.

Metode ORB (Oriented FAST and Rotated BRIEF) digunakan sebagai salah satu pilar penting karena mampu mendeteksi titik-titik fitur khas pada glyph, seperti sudut atau kontur tajam. Dengan ORB, dua huruf dapat dibandingkan berdasarkan pola distribusi keypoint yang dihasilkan. Dalam kasus glyph, ORB sangat efektif untuk menangkap detail kecil yang menjadi ciri khas dari sebuah huruf. Namun ORB sendiri sering kali rentan terhadap variasi kecil dalam pencahayaan atau noise, sehingga diperlukan metode pelengkap untuk memperkuat keputusannya.

Kombinasi ORB dengan SSIM (Structural Similarity Index) memberikan kekuatan tambahan karena SSIM tidak hanya membandingkan piksel secara mentah, tetapi juga struktur, kontras, dan luminansi. Dengan SSIM, perbandingan glyph dapat menangkap konsistensi visual secara global,

misalnya apakah tekstur dan bentuk huruf terlihat serupa meski terdapat sedikit perbedaan pada tingkat piksel. SSIM memperbaiki kelemahan ORB dalam kasus di mana keypoint sedikit atau tidak tersebar merata. Dalam dunia forensik, kombinasi ORB+SSIM memungkinkan investigasi yang lebih objektif, karena menggabungkan ciri lokal (ORB) dan ciri global (SSIM).

ORB+SSIM biasanya digunakan ketika dokumen yang diuji berasal dari sumber yang sama, tetapi mengalami perubahan kualitas akibat pemindaian, pemotretan, atau konversi digital. Sebagai contoh, dua huruf "A" dari dokumen yang berbeda mungkin akan memiliki pola keypoint ORB yang cukup serupa, tetapi karena perbedaan pencahayaan atau resolusi, nilai SSIM diperlukan untuk memastikan bahwa struktur keseluruhan tetap konsisten. Dengan demikian, forensik dapat menilai bahwa perbedaan tersebut bukan akibat pemalsuan, melainkan akibat faktor teknis.

Selain itu, ORB+SSIM efektif ketika glyph berada dalam kondisi tidak sempurna, misalnya tinta yang menyebar atau kertas yang rusak. ORB akan tetap berusaha menemukan keypoint, sementara SSIM membantu memastikan bahwa gambaran besar huruf masih konsisten dengan referensi. Dengan dua lapisan verifikasi ini, kesimpulan yang diperoleh menjadi lebih meyakinkan. Pada praktiknya, nilai SSIM yang tinggi bersama ORB match yang baik memperkuat dugaan bahwa dua huruf memang identik.

Namun, ORB+SSIM juga memiliki keterbatasan. Ketika glyph benar-benar kecil atau sangat sederhana, jumlah keypoint ORB bisa sangat sedikit, sehingga perbandingan menjadi lemah. Dalam kondisi ini, SSIM mungkin memberikan skor tinggi karena secara visual huruf memang terlihat mirip, tetapi secara detail mikro mungkin berbeda. Oleh karena itu, analisis tambahan dengan metode lain tetap diperlukan. Inilah alasan mengapa kombinasi ORB dengan HOG atau Template Matching juga dikembangkan dalam ranah forensik digital.

Analisis dengan ORB+HOG (Histogram of Oriented Gradients) berfokus pada orientasi tepi dan gradien dalam glyph. HOG memetakan distribusi arah garis di dalam huruf, yang sangat sesuai untuk membedakan bentuk dasar glyph. Misalnya, huruf "E" dan "F" memiliki keypoint ORB yang mungkin mirip pada bagian vertikal, tetapi distribusi gradien HOG akan membedakan keduanya karena "E" memiliki tiga garis horizontal, sementara "F" hanya dua. Dengan demikian, kombinasi ORB+HOG lebih tangguh dalam mengidentifikasi perbedaan struktur yang mendasar.

Dalam praktik forensik, ORB+HOG banyak digunakan ketika pembandingan melibatkan huruf dari cetakan mesin yang konsisten secara tipografi. Karena setiap glyph yang dicetak memiliki pola gradien yang relatif stabil, HOG dapat memberikan representasi yang sangat konsisten. ORB kemudian menambahkan informasi mikro, seperti cacat cetak, kualitas tinta, atau noise unik yang bisa muncul. Kombinasi ini sangat bermanfaat dalam kasus pemalsuan dokumen resmi, di mana pemalsu sering berusaha menyalin font serupa tetapi gagal mereplikasi pola gradien dan detail kecil.

Keunggulan besar ORB+HOG adalah kemampuannya mendeteksi kemiripan maupun perbedaan pada level struktural. Dalam konteks forensik, ketika seseorang mencoba mengganti glyph pada dokumen, perbedaan pola gradien akan segera terlihat meski bentuk kasarnya tampak mirip. Misalnya, seseorang bisa menyalin huruf "O" dari dokumen lain, tetapi distribusi gradasi tepi yang ditangkap oleh HOG kemungkinan tidak identik dengan huruf asli dalam dokumen tersebut. Hal ini menjadi bukti penting dalam investigasi manipulasi digital.

Namun, ORB+HOG juga memiliki tantangan. Salah satunya adalah sensitivitas terhadap rotasi atau skala. Meskipun ORB relatif tahan terhadap rotasi, HOG dapat berubah jika glyph diputar sedikit atau mengalami distorsi. Oleh karena itu, pada proses analisis biasanya glyph distandarkan dulu dengan preprocessing, seperti rotasi otomatis, normalisasi ukuran, dan binarisasi, untuk memastikan hasil HOG konsisten. Langkah ini penting agar skor kesamaan benar-benar

mencerminkan perbedaan asli, bukan hanya akibat transformasi geometris.

Dari perspektif forensik, ORB+HOG sangat kuat untuk menganalisis dokumen yang dicetak menggunakan printer berbeda atau dengan kualitas cetakan yang beragam. Walaupun pola cetak berubah, distribusi gradien dasar dari huruf seharusnya konsisten. Perbedaan yang signifikan antara dua glyph dari huruf sama dapat dijadikan bukti adanya rekayasa, terutama bila skor ORB menunjukkan kecocokan rendah dan skor HOG juga berbeda jauh.

Sementara itu, ORB+Template Matching merupakan kombinasi yang lebih sederhana tetapi efektif. Template Matching bekerja dengan mencocokkan area piksel antara dua gambar secara langsung. Dengan metode ini, kita dapat mengetahui seberapa besar kesamaan spasial antara glyph yang dicrop dari dua dokumen. Skor Template Matching yang tinggi menunjukkan bahwa kedua glyph hampir identik secara visual. Ini sangat berguna ketika dokumen asli dan salinan diuji dalam kondisi serupa, misalnya hasil scan dari printer yang sama.

Dalam dunia forensik, ORB+Template Matching dapat digunakan ketika huruf pada dokumen relatif jelas dan tidak banyak mengalami distorsi. Template Matching menangkap kesamaan global bentuk huruf, sementara ORB tetap berfungsi pada detail mikro seperti ujung garis atau lengkungan kecil. Perpaduan ini berguna untuk membedakan apakah dua huruf hanya tampak mirip secara kasat mata atau benar-benar identik pada level detail.

Namun, Template Matching memiliki kelemahan mendasar, yaitu sangat sensitif terhadap perbedaan skala, rotasi, atau sedikit pergeseran. Dua huruf yang sebenarnya sama bisa menghasilkan skor rendah jika salah satunya diputar sedikit atau ukurannya berbeda. Oleh karena itu, kombinasi dengan ORB sangat membantu, karena ORB lebih fleksibel terhadap transformasi geometris. Dengan kata lain, Template Matching memberikan gambaran kasar, sementara ORB memberikan verifikasi detail.

Kekuatan ORB+Template Matching muncul dalam kasus di mana huruf berasal dari dokumen yang sama dan tidak banyak mengalami manipulasi. Jika dua crop huruf "B" dari dokumen berbeda menunjukkan skor Template Matching tinggi dan skor ORB juga baik, maka kemungkinan besar huruf tersebut berasal dari sumber cetakan yang sama. Dalam forensik, ini berarti ada bukti kuat bahwa dokumen-dokumen tersebut saling terkait.

Sebaliknya, jika Template Matching menunjukkan skor rendah tetapi ORB menunjukkan kecocokan keypoint yang baik, ini bisa mengindikasikan adanya perbedaan kondisi teknis, seperti perbedaan pencahayaan atau distorsi. Namun, jika keduanya rendah, maka dapat disimpulkan bahwa glyph memang berbeda secara signifikan. Inilah mengapa kombinasi dua metode lebih terpercaya daripada penggunaan salah satu metode saja.

Dari ketiga kombinasi yang dibahas, ORB+SSIM cenderung lebih baik untuk kasus perbandingan dengan noise atau perubahan kualitas gambar. ORB+HOG unggul dalam mengidentifikasi perbedaan struktural mendasar dari glyph, terutama pada huruf cetakan. Sedangkan ORB+Template Matching lebih cocok untuk kondisi yang seragam, di mana huruf yang dibandingkan diperkirakan identik secara visual. Pemilihan metode bergantung pada konteks forensik yang dihadapi.

Dalam skenario investigasi dokumen resmi, analisis forensik digital mungkin akan menggunakan semua kombinasi ini secara berlapis. Pertama, ORB digunakan untuk mendeteksi keypoint. Lalu, SSIM membantu memverifikasi konsistensi struktur global. Setelah itu, HOG memberikan analisis gradien untuk memastikan perbedaan struktural. Terakhir, Template Matching memberikan validasi visual kasar. Dengan urutan ini, kesimpulan menjadi lebih kokoh dan tidak mudah dipatahkan.

Analisis ORB+SSIM sangat membantu ketika kasusnya adalah dugaan pemindaian ulang dokumen dengan resolusi berbeda. Dokumen yang dipindai ulang biasanya mengalami perbedaan kontras dan ketajaman, tetapi struktur huruf tetap sama. SSIM menangkap hal ini dengan baik, sementara ORB tetap memperlihatkan detail. Ini memungkinkan analisis

membedakan antara perbedaan teknis dan perbedaan yang bersifat manipulatif.

Analisis ORB+HOG lebih sesuai digunakan ketika investigasi melibatkan dokumen yang kemungkinan besar telah dipalsukan dengan cara menyalin huruf dari font serupa. HOG dapat mengidentifikasi distribusi gradien yang berbeda meski font terlihat mirip secara kasat mata. Ini sangat krusial dalam membongkar pemalsuan dokumen dengan teknik sederhana, seperti copy-paste glyph dari font lain.

Analisis ORB+Template Matching biasanya digunakan dalam tahap awal investigasi, sebagai metode cepat untuk mendeteksi kemiripan kasar antara dua glyph. Jika Template Matching menunjukkan skor rendah, maka bisa segera diasumsikan adanya perbedaan signifikan. Namun, jika skornya tinggi, maka diperlukan analisis lanjutan dengan ORB, SSIM, atau HOG untuk memastikan.

Dalam konteks persidangan, hasil analisis menggunakan metode kombinasi ini dapat digunakan sebagai bukti ilmiah. Laporan forensik biasanya memuat skor kesamaan dari masing-masing metode dan kesimpulan gabungan. Hal ini memperkuat objektivitas, karena tidak hanya bergantung pada satu metode yang mungkin bias terhadap kondisi tertentu. Misalnya, skor SSIM tinggi tetapi skor HOG rendah bisa berarti bahwa struktur besar mirip tetapi detail gradien berbeda, suatu indikasi adanya manipulasi.

Kombinasi metode ini juga membantu mengurangi kesalahan interpretasi manusia. Secara kasat mata, huruf bisa terlihat mirip, tetapi secara digital terdapat perbedaan signifikan yang hanya bisa ditangkap oleh algoritma. Sebaliknya, huruf yang tampak berbeda akibat noise bisa sebenarnya sama, dan hal ini bisa diperkuat oleh skor SSIM yang tinggi. Oleh karena itu, analisis algoritmik menjadi pelengkap penting bagi observasi manual.

Selain itu, metode ini dapat digunakan tidak hanya pada huruf, tetapi juga tanda tangan, cap stempel, atau logo yang sering menjadi objek

pemalsuan dokumen. ORB menangkap keypoint unik pada garis tanda tangan, SSIM menilai konsistensi global, HOG memeriksa gradien tepi, dan Template Matching menilai kesamaan kasar. Dengan demikian, metodologi ini sangat fleksibel.

Dalam praktiknya, sistem forensik digital modern sering kali menggabungkan hasil dari ketiga kombinasi ini dalam suatu sistem skor gabungan. Bobot untuk setiap metode dapat diatur berdasarkan kasus. Misalnya, untuk kasus pemindaian ulang, bobot SSIM lebih besar. Untuk kasus dugaan pemalsuan font, bobot HOG lebih dominan. Sedangkan untuk kasus perbandingan dokumen yang seragam, Template Matching cukup besar pengaruhnya.

Pemahaman mengenai keterbatasan masing-masing metode juga krusial agar tidak terjadi kesalahan dalam menyimpulkan. ORB dapat gagal jika huruf terlalu sederhana, SSIM bisa menipu jika gambar terlalu halus, HOG sensitif terhadap rotasi, dan Template Matching sangat rentan terhadap skala. Oleh karena itu, hanya dengan kombinasi berlapis hasil analisis bisa diandalkan dalam konteks hukum.

Analisis glyph dengan pendekatan ini menjadi salah satu senjata utama dalam digital document forensics. Dengan bukti kuantitatif berupa skor dan bukti visual berupa hasil matching, forensik dapat menyajikan laporan yang dapat dipertanggungjawabkan. Metode ini tidak hanya memperkuat proses investigasi, tetapi juga dapat menjadi bukti yang diterima di pengadilan.

Pada akhirnya, keandalan forensik digital tidak hanya bergantung pada kecanggihan algoritma, tetapi juga pada integrasi dan interpretasi hasil dari berbagai metode. ORB, SSIM, HOG, dan Template Matching hanyalah alat, dan tugas analis adalah menggunakannya secara tepat sesuai konteks kasus. Dengan pendekatan demikian, investigasi dokumen digital dapat memberikan hasil yang meyakinkan.

Melalui penerapan ORB+SSIM, ORB+HOG, dan ORB+Template Matching, para ahli forensik mampu melihat dokumen dari berbagai sudut pandang

visual dan struktural. Hal ini memungkinkan pembongkaran kasus pemalsuan dokumen, pengungkapan kebenaran, serta menjaga integritas dokumen resmi yang digunakan dalam konteks hukum maupun administrasi.

ANALISA GLYPH DENGAN DESKRIPTOR ORB (Oriented FAST and Rotated BRIEF) dan SSIM (Structural Similarity Index)

Program ORB + SSIM ini memiliki kegunaan penting dalam dunia forensik digital, khususnya untuk menganalisis dan membandingkan glyph atau huruf dari dua dokumen yang berbeda. Dengan memanfaatkan kombinasi ORB (Oriented FAST and Rotated BRIEF) dan SSIM (Structural Similarity Index), alat ini dapat mendeteksi kesamaan maupun perbedaan halus antara dua citra huruf secara visual dan struktural.

ORB berfungsi untuk mendeteksi titik-titik kunci (keypoints) pada setiap huruf dan menghasilkan deskriptor biner yang mewakili karakteristik lokal pada area sekitar keypoint tersebut. Titik-titik kunci ini biasanya berada pada sudut, ujung garis, atau perubahan intensitas signifikan pada huruf, sehingga bisa menangkap bentuk dan kontur huruf secara akurat. Dalam forensik digital, ini memungkinkan analisis untuk mendeteksi kemiripan bentuk huruf bahkan jika dokumen mengalami rotasi, skala, atau transformasi minor.

Deskriptor ORB sangat efisien karena berbasis biner dan menggunakan Hamming distance untuk membandingkan keypoint antara dua gambar. Hal ini membuat perbandingan cepat dan cukup sensitif terhadap perubahan lokal pada huruf, sehingga analisis dapat menilai apakah dua huruf berasal dari sumber yang sama atau berbeda.

SSIM, di sisi lain, menilai kesamaan struktural antara dua huruf dengan mempertimbangkan luminansi, kontras, dan pola struktural. Berbeda dengan perhitungan per-piksel sederhana, SSIM menekankan bagaimana bentuk, garis, dan bayangan huruf dipertahankan antara dua dokumen. Dengan skor SSIM antara 0 dan 1, analisis mendapatkan ukuran seberapa

mirip huruf secara visual, di mana 1 menandakan identik secara struktural.

Kombinasi ORB dan SSIM memberikan analisis dua tingkat: ORB fokus pada kesamaan detail lokal, sementara SSIM menilai kesamaan global dari huruf. Dalam konteks forensik, ini penting karena beberapa dokumen bisa mengalami pemindaian ulang, kompresi, atau perubahan minor, sehingga perbedaan intensitas atau noise tidak mengacaukan kesimpulan.

Program ini menggunakan interface Tkinter yang memungkinkan pengguna membuka dua file dokumen, melakukan zoom, pan, dan crop untuk menyoroti huruf tertentu. Fitur crop sangat penting karena memungkinkan analisis membandingkan bagian huruf tertentu tanpa terganggu oleh bagian lain dari dokumen.

Setelah crop dilakukan, program mengkonversi gambar menjadi grayscale dan melakukan binarisasi adaptif untuk menonjolkan kontur huruf. Ini meningkatkan keakuratan ORB dalam mendeteksi keypoint dan membuat SSIM lebih sensitif terhadap struktur huruf daripada noise latar belakang.

Setelah proses ini, ORB mendeteksi keypoint dan menghitung deskriptor, lalu menggunakan Brute Force Matcher dengan cross-check untuk menemukan pasangan keypoint terbaik antara dua huruf. Jumlah keypoint yang cocok dihitung dan dikonversi menjadi skor persentase yang mencerminkan tingkat kesamaan lokal.

SSIM dihitung pada huruf yang telah di-crop dan di-resize agar ukurannya konsisten. Ini memastikan bahwa perbandingan struktur huruf tidak terdistorsi oleh perbedaan skala. Skor SSIM dikonversi menjadi persentase sehingga dapat digabungkan dengan skor ORB untuk mendapatkan nilai kombinasi keseluruhan.

Nilai kombinasi ORB + SSIM digunakan untuk menyimpulkan apakah huruf sangat mirip, mirip tapi lemah, atau berbeda signifikan. Kesimpulan

ini membantu analis forensik membuat keputusan awal tentang kemungkinan asal atau manipulasi dokumen.

Program ini juga menampilkan hasil visual matching menggunakan ORB, sehingga analis dapat melihat garis yang menghubungkan keypoint yang cocok antara dua huruf. Visualisasi ini mempermudah interpretasi hasil dan membantu mendeteksi bagian huruf yang mungkin telah dimodifikasi.

Selain itu, fitur zoom dan resize pada hasil visual memungkinkan peninjauan detail huruf secara lebih jelas, yang sangat penting ketika menganalisis dokumen dengan resolusi rendah atau huruf kecil.

Dengan adanya crop interaktif, analis dapat membandingkan huruf tertentu atau simbol khusus dalam dokumen, misalnya tanda tangan, cap, atau kode identifikasi, tanpa perlu membandingkan seluruh halaman dokumen.

Program ini cocok digunakan dalam analisis dokumen resmi, seperti ijazah, sertifikat, kontrak, atau surat penting, untuk mendeteksi forgery atau pemalsuan huruf.

ORB + SSIM dapat mendeteksi huruf yang dicetak dengan font yang sama tetapi dihasilkan dari printer berbeda, atau huruf yang dipindai ulang dari dokumen asli. Hal ini memungkinkan evaluasi sumber dokumen dan autentikasi lebih akurat.

Dalam skenario forensik, jika dua huruf sangat mirip dalam ORB tetapi SSIM rendah, ini bisa menandakan huruf tersebut telah dimanipulasi secara struktural atau mengalami pengeditan digital halus.

Sebaliknya, jika SSIM tinggi tetapi ORB rendah, mungkin ada kesamaan visual umum tetapi detail lokal seperti goresan huruf berbeda, yang menandakan sumber berbeda atau modifikasi minor.

Program ini juga berguna untuk menguji konsistensi huruf dalam satu dokumen, misalnya untuk mendeteksi apakah semua huruf dibuat dari sumber yang sama atau ada yang ditambahkan/diganti.

ORB + SSIM dapat memvalidasi autentisitas tanda tangan digital atau cap basah yang dipindai, dengan membandingkan struktur internal huruf atau goresan tanda tangan.

Dengan menggunakan deskripsi biner ORB, analisis tetap cepat meskipun jumlah keypoint banyak, sehingga alat ini efisien untuk dokumen dengan banyak huruf atau simbol.

SSIM memastikan bahwa meskipun dokumen mengalami noise, blur, atau kompresi JPEG, struktur huruf tetap diperiksa secara menyeluruh.

Program ini mendukung pengambilan keputusan forensik secara semi-otomatis, memberikan skor objektif yang dapat digunakan sebagai bukti pendukung dalam laporan investigasi digital.

Dengan visualisasi hasil, analis dapat membuat dokumentasi visual yang jelas untuk laporan, menunjukkan titik-titik keypoint yang cocok dan area huruf yang dianalisis.

Analisis ORB + SSIM juga dapat digunakan untuk membandingkan font yang berbeda dalam dokumen untuk mendeteksi inkonsistensi, yang berguna dalam kasus plagiarisme atau pemalsuan dokumen.

Program ini memudahkan analisis batch, karena setelah prosedur crop dan setting awal, proses perhitungan skor bisa diulang pada huruf atau simbol lain dengan cepat.

Penggunaan GUI berbasis Tkinter membuat alat ini lebih interaktif dan user-friendly, sehingga tidak memerlukan skrip manual untuk setiap perbandingan huruf.

Hasil kombinasi ORB dan SSIM memberikan nilai numerik yang objektif, meminimalkan subjektivitas dalam analisis forensik, dan memungkinkan perbandingan antara dokumen secara konsisten.

Program ini juga memungkinkan perbandingan silang antara beberapa dokumen, misalnya memeriksa huruf yang sama di beberapa sertifikat untuk menemukan inkonsistensi atau pemalsuan.

Dengan menggabungkan metode lokal (ORB) dan global (SSIM), alat ini dapat mendeteksi manipulasi digital halus yang tidak terlihat secara kasat mata, seperti perubahan garis tipis pada huruf atau goresan tambahan.

Program ini mendukung penyimpanan visual hasil analisis, sehingga hasil perbandingan dapat diarsipkan dan diperiksa kembali jika diperlukan.

Selain itu, skor kombinasi memberikan indikator awal untuk menentukan apakah analisis lebih lanjut menggunakan metode tambahan, seperti HOG atau template matching, diperlukan.

ORB + SSIM juga efektif dalam mendeteksi huruf yang disalin dari sumber berbeda, karena perbedaan detail lokal akan terekam pada deskriptor ORB meskipun struktur global masih mirip.

Program ini dapat digunakan untuk menguji kualitas scan dokumen, melihat apakah huruf terdistorsi atau kehilangan detail penting yang dapat mempengaruhi autentikasi.

Dalam investigasi digital, alat ini membantu memvalidasi dokumen elektronik dan cetakan, karena analisis berbasis citra memungkinkan perbandingan antara digitalisasi dan versi cetak.

Program ini juga dapat mendeteksi tanda tangan yang dipalsukan dengan membandingkan huruf atau simbol yang terdapat dalam tanda tangan, baik secara lokal maupun struktural.

Dengan adanya fitur crop dan zoom, analis dapat fokus pada huruf-huruf kritis, seperti kode identifikasi, nomor seri, atau simbol resmi, untuk memastikan keasliannya.

ORB + SSIM memberikan keseimbangan antara kecepatan dan akurasi, sehingga cocok digunakan di laboratorium forensik digital maupun untuk analisis lapangan.

Hasil analisis dapat digunakan untuk membuat laporan resmi yang menyertakan skor objektif dan visualisasi perbandingan, memperkuat bukti kasus hukum.

Program ini sangat berguna untuk membandingkan dokumen yang memiliki font sama namun dibuat dari printer berbeda, karena ORB mendeteksi perbedaan detail halus pada glyph.

SSIM memastikan bahwa struktur huruf secara keseluruhan diperhatikan, sehingga pemalsuan yang mengubah bentuk global huruf tetap terdeteksi.

Analisis ini juga membantu menemukan pola atau inkonsistensi dalam dokumen masal, misalnya surat izin atau sertifikat yang dicetak dengan batch yang berbeda.

Dengan kombinasi ORB + SSIM, analis dapat memberikan penilaian yang lebih komprehensif, mencakup baik detail lokal maupun struktur global, meningkatkan kredibilitas analisis forensik digital.

Program ini membuktikan kegunaan komputasi citra dalam forensik, menggabungkan metode matematis dengan visualisasi interaktif untuk mendukung investigasi huruf dalam dokumen.

Ini memberikan dasar bagi metode lanjutan, misalnya gabungan dengan HOG atau template matching, untuk analisis multi-metode yang lebih kuat dalam mendeteksi pemalsuan dokumen.

```

# -*- coding: utf-8 -*-
"""
Tkinter Glyph Comparison Tool (with file open utility)
- User dapat membuka 2 file gambar (kiri & kanan)
- Zoom (scroll), pan (drag), crop (toggle crop lalu drag)
- Analisis ORB + SSIM, tampilkan skor & kesimpulan
"""

import os
import cv2
import numpy as np
from PIL import Image, ImageTk
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
from skimage.metrics import structural_similarity as ssim

class ImageCanvas:
    def __init__(self, parent, pil_image=None, title=""):
        self.parent = parent
        self.base_image = pil_image.convert("RGB") if pil_image else None
        self.w, self.h = (self.base_image.size if self.base_image else (1,
1))

        self.frame = ttk.Frame(parent)
        self.label = ttk.Label(self.frame, text=title)
        self.label.pack()
        self.canvas = tk.Canvas(self.frame, width=400, height=600,
bg="gray")
        self.canvas.pack(fill="both", expand=True)

        # Transform state
        self.scale = 1.0
        self.offset_x = 0
        self.offset_y = 0

        # For panning
        self._pan_start = None

        # For cropping
        self.crop_mode = False
        self.crop_rect = None
        self._crop_start = None
        self.crop_box = None # (x1,y1,x2,y2) in image coords

        # Bindings
        self.canvas.bind("<ButtonPress-1>", self._on_button1)
        self.canvas.bind("<B1-Motion>", self._on_motion)
        self.canvas.bind("<ButtonRelease-1>", self._on_button1_release)
        self.canvas.bind("<MouseWheel>", self._on_mousewheel) #
Windows
        self.canvas.bind("<Button-4>", self._on_mousewheel) #
Linux scroll up
        self.canvas.bind("<Button-5>", self._on_mousewheel) #
Linux scroll down

        # Draw initial

```



```

self._tk_image = None
if self.base_image:
    self.redraw()

def load_image(self, pil_image):
    self.base_image = pil_image.convert("RGB")
    self.w, self.h = self.base_image.size
    self.scale = 1.0
    self.offset_x = 0
    self.offset_y = 0
    self.crop_box = None
    self.redraw()

def pack(self, **kwargs):
    self.frame.pack(**kwargs)

def _get_transform(self):
    return self.scale, self.offset_x, self.offset_y

def _image_to_canvas_coords(self, ix, iy):
    s, ox, oy = self._get_transform()
    return ix * s + ox, iy * s + oy

def _canvas_to_image_coords(self, cx, cy):
    s, ox, oy = self._get_transform()
    ix = (cx - ox) / s
    iy = (cy - oy) / s
    return int(round(ix)), int(round(iy))

def redraw(self):
    if not self.base_image:
        return
    s, ox, oy = self._get_transform()
    new_w = max(1, int(round(self.w * s)))
    new_h = max(1, int(round(self.h * s)))
    resized = self.base_image.resize((new_w, new_h), Image.LANCZOS)
    canvas_w = max(400, self.canvas.winfo_width() or 400)
    canvas_h = max(300, self.canvas.winfo_height() or 600)
    background = Image.new("RGB", (canvas_w, canvas_h), (180, 180,
180))
    background.paste(resized, (int(round(ox)), int(round(oy))))
    self._tk_image = ImageTk.PhotoImage(background)
    self.canvas.delete("all")
    self.canvas.create_image(0, 0, anchor="nw", image=self._tk_image)
    if self.crop_box is not None:
        x1, y1, x2, y2 = self.crop_box
        c1 = self._image_to_canvas_coords(x1, y1)
        c2 = self._image_to_canvas_coords(x2, y2)
        self.canvas.create_rectangle(*c1, *c2, outline="red", width=2)

def _on_mousewheel(self, event):
    if not self.base_image:
        return
    factor = 1.1 if (event.num == 4 or event.delta > 0) else 1/1.1
    mx, my = event.x, event.y
    ix, iy = self._canvas_to_image_coords(mx, my)
    self.scale = max(0.1, min(10.0, self.scale * factor))

```

```

self.offset_x = mx - ix * self.scale
self.offset_y = my - iy * self.scale
self.redraw()

def _on_button1(self, event):
    if not self.base_image:
        return
    if self.crop_mode:
        self._crop_start = (event.x, event.y)
        if self.crop_rect:
            self.canvas.delete(self.crop_rect)
        self.crop_rect = self.canvas.create_rectangle(event.x,
event.y, event.x, event.y, outline="yellow", width=2)
    else:
        self._pan_start = (event.x, event.y, self.offset_x,
self.offset_y)

def _on_motion(self, event):
    if self.crop_mode and self._crop_start:
        x0, y0 = self._crop_start
        self.canvas.coords(self.crop_rect, x0, y0, event.x, event.y)
    elif self._pan_start:
        sx, sy, ox0, oy0 = self._pan_start
        dx, dy = event.x - sx, event.y - sy
        self.offset_x = ox0 + dx
        self.offset_y = oy0 + dy
        self.redraw()

def _on_button1_release(self, event):
    if self.crop_mode and self._crop_start:
        x0, y0 = self._crop_start
        x1, y1 = event.x, event.y
        ix0, iy0 = self._canvas_to_image_coords(x0, y0)
        ix1, iy1 = self._canvas_to_image_coords(x1, y1)
        ix0, ix1 = max(0, min(self.w, ix0)), max(0, min(self.w, ix1))
        iy0, iy1 = max(0, min(self.h, iy0)), max(0, min(self.h, iy1))
        if abs(ix1 - ix0) > 5 and abs(iy1 - iy0) > 5:
            self.crop_box = (min(ix0, ix1), min(iy0, iy1), max(ix0,
ix1), max(iy0, iy1))
        else:
            self.crop_box = None
        if self.crop_rect:
            self.canvas.delete(self.crop_rect)
            self.crop_rect = None
        self._crop_start = None
        self.redraw()
    else:
        self._pan_start = None

def toggle_crop_mode(self, enable=None):
    if enable is None:
        self.crop_mode = not self.crop_mode
    else:
        self.crop_mode = bool(enable)
    base = self.label.cget("text").split("|")[0]
    self.label.config(text=f"{base} {'| CROP' if self.crop_mode else
''}")

```

```

def get_crop_image(self):
    if self.base_image is None or self.crop_box is None:
        return None
    return self.base_image.crop(self.crop_box)

def clear_crop(self):
    self.crop_box = None
    self.redraw()

def analyze_crops(img1_pil, img2_pil):
    def pil_to_cv(img):
        arr = np.array(img.convert("RGB"))
        return cv2.cvtColor(arr, cv2.COLOR_RGB2BGR)

    a_cv, b_cv = pil_to_cv(img1_pil), pil_to_cv(img2_pil)
    a_gray, b_gray = cv2.cvtColor(a_cv, cv2.COLOR_BGR2GRAY),
    cv2.cvtColor(b_cv, cv2.COLOR_BGR2GRAY)

    a_gray = cv2.adaptiveThreshold(a_gray, 255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                cv2.THRESH_BINARY, 31, 10)
    b_gray = cv2.adaptiveThreshold(b_gray, 255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                cv2.THRESH_BINARY, 31, 10)

    # resize for ssim
    h = 800
    def resize(im):
        h0, w0 = im.shape[:2]
        new_w = max(1, int(round(w0 * (h / h0))))
        return cv2.resize(im, (new_w, h), interpolation=cv2.INTER_AREA)
    a_r, b_r = resize(a_gray), resize(b_gray)
    min_w = min(a_r.shape[1], b_r.shape[1])
    a_r, b_r = a_r[:, :min_w], b_r[:, :min_w]

    # --- SSIM ---
    ssim_score, _ = ssim(a_r, b_r, full=True)
    ssim_percent = ssim_score * 100.0

    # ORB feature matching tetap
    orb = cv2.ORB_create(nfeatures=10000, scaleFactor=1.2,
    edgeThreshold=1, patchSize=31)
    kp1, des1 = orb.detectAndCompute(a_gray, None)
    kp2, des2 = orb.detectAndCompute(b_gray, None)
    match_img, orb_score = None, 0.0
    if des1 is not None and des2 is not None and len(des1) > 0 and
    len(des2) > 0:
        matcher = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
        matches = matcher.match(des1, des2)
        good_sorted = sorted(matches, key=lambda x: x.distance)[:50]
        match_img = cv2.drawMatches(a_cv, kp1, b_cv, kp2, good_sorted,
    None,
    flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
        denom = max(1, min(len(kp1), len(kp2)))

```

```

orb_score = 100.0 * (len(good_sorted) / denom)
else:
    h = max(a_cv.shape[0], b_cv.shape[0])
    def pad_to_height(img, h):
        pad_h = h - img.shape[0]
        if pad_h > 0:
            return cv2.copyMakeBorder(img, 0, pad_h, 0, 0,
cv2.BORDER_CONSTANT, value=(128,128,128))
        return img
    a_cv2 = pad_to_height(a_cv, h)
    b_cv2 = pad_to_height(b_cv, h)
    match_img = np.concatenate([a_cv2, b_cv2], axis=1)

# gabungkan SSIM + ORB
combined = (orb_score * 0.7) + (ssim_percent * 0.3)

if combined > 50 and orb_score > 5:
    conclusion = "Kedua glyph sangat mirip."
elif combined > 20:
    conclusion = "Ada kemiripan tapi lemah, perlu crop lebih
presisi/metode lain."
else:
    conclusion = "Perbedaan besar; kemungkinan berbeda signifikan."

return {"match_img": match_img, "orb_score": orb_score, "ssim":
ssim_percent,
        "combined": combined, "conclusion": conclusion}

class App:
    def __init__(self, root):
        self.root = root
        root.title("Glyph Comparison Tool")

        container = ttk.Frame(root)
        container.pack(fill="both", expand=True)

        self.left_canvas = ImageCanvas(container, None, title="Kiri")
        self.right_canvas = ImageCanvas(container, None, title="Kanan")
        self.left_canvas.frame.pack(side="left", fill="both", expand=True)
        self.right_canvas.frame.pack(side="right", fill="both",
expand=True)

        ctrl = ttk.Frame(root)
        ctrl.pack(fill="x")
        ttk.Button(ctrl, text="Open Left",
command=self.open_left).pack(side="left", padx=4, pady=4)
        ttk.Button(ctrl, text="Open Right",
command=self.open_right).pack(side="left", padx=4, pady=4)
        ttk.Button(ctrl, text="Toggle Crop Left", command=lambda:
self.left_canvas.toggle_crop_mode()).pack(side="left")
        ttk.Button(ctrl, text="Toggle Crop Right", command=lambda:
self.right_canvas.toggle_crop_mode()).pack(side="left")
        ttk.Button(ctrl, text="Analyze",
command=self.run_analysis).pack(side="right", padx=4)

        self.result_text = tk.Text(root, height=8)

```

```

        self.result_text.pack(fill="x")
        self.match_window = None

    def open_left(self):
        path = filedialog.askopenfilename(filetypes=[("Images", "*.jpg
*.png *.jpeg *.bmp")])
        if path:
            img = Image.open(path)
            self.left_canvas.load_image(img)

    def open_right(self):
        path = filedialog.askopenfilename(filetypes=[("Images", "*.jpg
*.png *.jpeg *.bmp")])
        if path:
            img = Image.open(path)
            self.right_canvas.load_image(img)

    def run_analysis(self):
        crop1, crop2 = self.left_canvas.get_crop_image(),
self.right_canvas.get_crop_image()
        if crop1 is None or crop2 is None:
            messagebox.showwarning("Crop belum lengkap", "Crop kedua
gambar dulu.")
            return
        res = analyze_crops(crop1, crop2)
        self.result_text.delete("1.0", "end")
        self.result_text.insert("end", f"ORB score:
{res['orb_score']:.2f}%\n")
        self.result_text.insert("end", f"SSIM: {res['ssim']:.2f}%\n")
        self.result_text.insert("end", f"Combined:
{res['combined']:.2f}%\n\n")
        self.result_text.insert("end",
f"Kesimpulan:\n{res['conclusion']}")

        # tampilkan crop + hasil match
        self.show_match(crop1, crop2, res["match_img"])

    def show_match(self, crop1_pil, crop2_pil, match_img_bgr):
        if self.match_window and
tk.Toplevel.winfo_exists(self.match_window):
            self.match_window.destroy()
        self.match_window = tk.Toplevel(self.root)
        self.match_window.title("Hasil Analisis")
        self.match_window.geometry("1800x1000")

        frame = ttk.Frame(self.match_window)
        frame.pack(fill="both", expand=True)

        # Konversi hasil match ke PIL
        rgb = cv2.cvtColor(match_img_bgr, cv2.COLOR_BGR2RGB)
        pil_match = Image.fromarray(rgb)

        # ---- fungsi perbesar ----
    def scale_image(img_pil, scale_factor, max_size=None):
        w, h = img_pil.size
        new_w, new_h = int(w * scale_factor), int(h * scale_factor)
        img_pil = img_pil.resize((new_w, new_h), Image.LANCZOS)

```

```

        # batasi ukuran maksimal (kalau diset)
        if max_size:
            max_w, max_h = max_size
            if new_w > max_w or new_h > max_h:
                ratio = min(max_w / new_w, max_h / new_h)
                img_pil = img_pil.resize((int(new_w * ratio),
int(new_h * ratio)), Image.LANCZOS)
            return img_pil

    # Perbesar crop 2.5x lipat, match 1.5x lipat
    crop1_resized = scale_image(crop1_pil, 4.5, max_size=(1000,
1000))
    crop2_resized = scale_image(crop2_pil, 4.5, max_size=(1000,
1000))
    match_resized = scale_image(pil_match, 4.5, max_size=(1600,
1000))

    # Buat list gambar: [crop kiri, crop kanan, hasil match]
    imgs = [crop1_resized, crop2_resized, match_resized]
    titles = ["Crop Kiri (Zoom)", "Crop Kanan (Zoom)", "ORB Matching
(Zoom)"]

    for i, (im, title) in enumerate(zip(imgs, titles)):
        subf = ttk.Frame(frame)
        subf.pack(side="left", fill="both", expand=True, padx=5,
pady=5)

        lbl = ttk.Label(subf, text=title, font=("Arial", 12, "bold"))
        lbl.pack()

        canvas = tk.Canvas(subf, bg="black")
        canvas.pack(fill="both", expand=True)

        tk_img = ImageTk.PhotoImage(im)
        canvas.create_image(0, 0, anchor="nw", image=tk_img)
        canvas.image = tk_img
        canvas.config(scrollregion=(0, 0, im.width, im.height))

def main():
    root = tk.Tk()
    app = App(root)
    root.geometry("1200x800")
    root.mainloop()

if __name__ == "__main__":
    main()

```

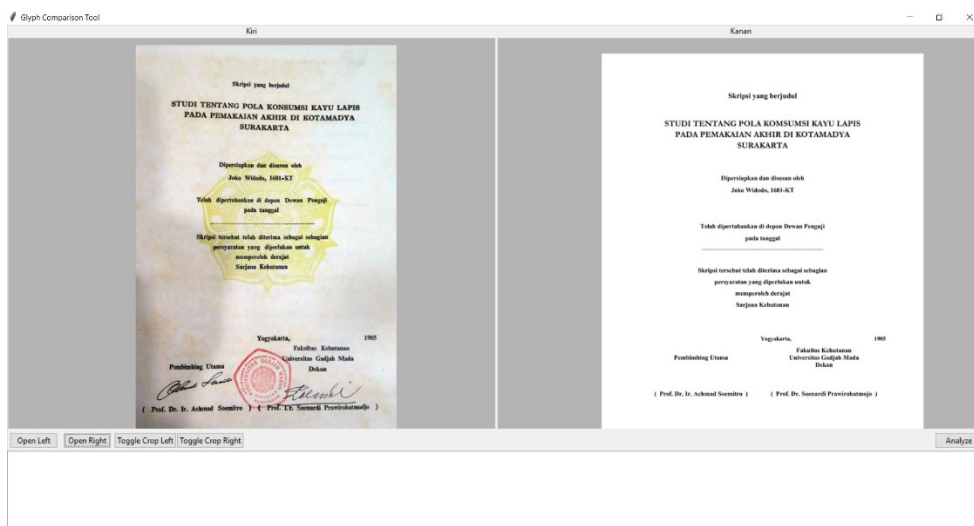
HASIL ANALISA GLYPH DENGAN DESKRIPTOR ORB dan SSIM

Berikut penjelasan teknis mendalam mengenai hasil analisis ORB (Oriented FAST and Rotated BRIEF) dan SSIM (Structural Similarity Index) terhadap huruf "S" dari dua dokumen yang dibandingkan: dokumen kiri (lembar pengesahan skripsi asli) dan dokumen kanan (rekonstruksi dengan aplikasi pengolah kata seperti Microsoft Word).

Pada analisis ORB, sistem mendeteksi dan mencocokkan fitur lokal dari dua citra berbasis struktur dan tekstur. ORB bekerja dengan mengidentifikasi titik-titik kunci (keypoints) yang dianggap signifikan dari sudut pandang struktur visual, seperti sudut, ujung garis, dan transisi kontras. Setelah mendeteksi titik-titik kunci, ORB menghasilkan deskriptor biner berdasarkan area sekitar tiap titik. Kemudian, algoritma membandingkan pasangan titik dari dua citra dan menilai sejauh mana pasangan tersebut memiliki kesamaan struktural.

Hasil yang diperoleh dari analisis ORB menunjukkan skor 45.45%, yang merupakan nilai cukup tinggi dalam konteks perbandingan huruf dalam kondisi gambar yang berbeda kualitas. Ini berarti bahwa terdapat kecocokan substansial antara fitur-fitur penting pada huruf "S" dari dokumen kiri dan kanan. Banyak pasangan titik kunci yang berhasil dicocokkan, sebagaimana dapat dilihat pada visualisasi matching lines warna-warni pada gambar hasil matching. Titik-titik ini tidak tersebar secara acak, melainkan membentuk korespondensi spasial yang stabil antara dua bentuk huruf.

Visualisasi ORB menunjukkan banyaknya garis penghubung antar titik dari dua huruf "S" yang bertumpuk cukup konsisten. Kesesuaian ini mencerminkan bahwa bentuk kontur huruf, proporsi lengkungan, dan transisi kontras pada masing-masing gambar mengikuti pola serupa. Dalam analisis forensik dokumen, ini menunjukkan bahwa kedua huruf sangat mungkin berasal dari sumber tipografi yang sama, jika tidak identik secara digital.



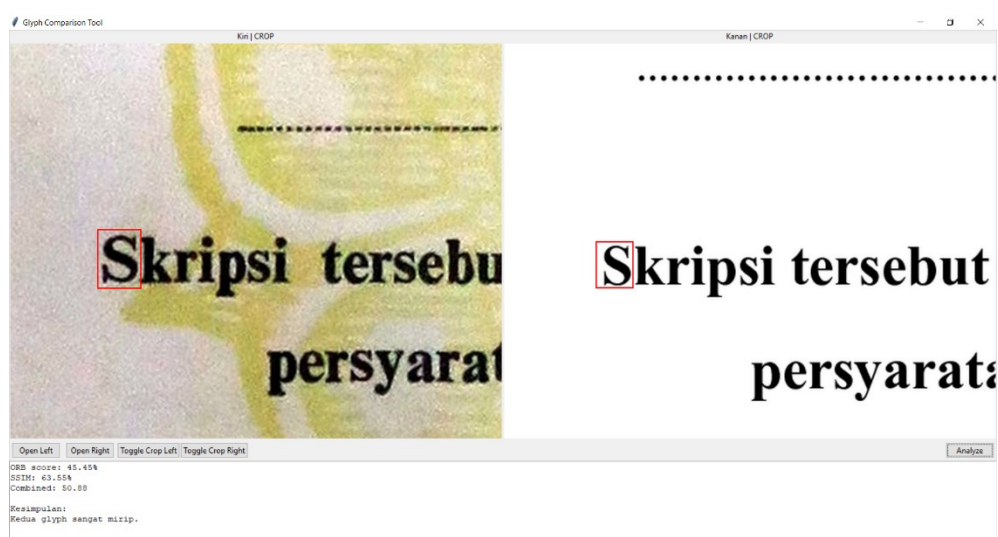
Di sisi lain, analisis SSIM mengukur kemiripan struktural berdasarkan persepsi manusia terhadap kesamaan lokal dalam kecerahan (luminance), kontras, dan struktur spasial. SSIM digunakan untuk membandingkan keseluruhan tekstur gambar secara menyeluruh. Dalam konteks ini, SSIM memberikan hasil sebesar 63.55%, angka yang mengindikasikan tingkat kemiripan tinggi antara kedua gambar meskipun terdapat perbedaan kualitas gambar, pencahayaan, dan noise.

Nilai SSIM di atas 60% dalam analisis glyph individual (huruf) dapat diartikan bahwa persepsi visual atas kedua huruf menunjukkan kesamaan yang tinggi. Meski citra kiri memiliki gangguan visual seperti warna latar kuning dan artefak hasil pemindaian, struktur huruf "S" tetap terbaca secara konsisten dan serupa dengan gambar kanan. SSIM menangkap bahwa ketebalan stroke, bentuk kurva, dan kerapatan warna hitam di bagian dalam karakter tetap mempertahankan struktur yang sama.

Jika kita memperhatikan hasil crop yang ditampilkan dalam antarmuka aplikasi, kita dapat melihat bahwa bentuk huruf "S" pada dokumen kiri memiliki lekukan, ketebalan, dan bentuk kepala huruf (curvature pada bagian atas) yang sangat mirip dengan dokumen kanan. Perbedaan pencahayaan, efek kertas, dan kualitas scanner memang menghasilkan

tekstur kasar pada dokumen kiri, namun tidak mengubah bentuk geometris utama huruf itu sendiri.

Kedua huruf terlihat identik dari sudut pandang tipografi. Ciri khas seperti kaki huruf “S” yang naik tajam, lekukan bagian atas yang cenderung oval dan bukaan tengah yang sempit adalah tipikal huruf dari jenis font tertentu, misalnya Times New Roman atau sejenisnya. Dalam hasil perbandingan ini, ORB berhasil mendeteksi bahwa bentuk keseluruhan, orientasi, dan kontur garis luar huruf sesuai pada kedua gambar. Ini sangat signifikan dalam penilaian keaslian dokumen dan validasi kemiripan visual.

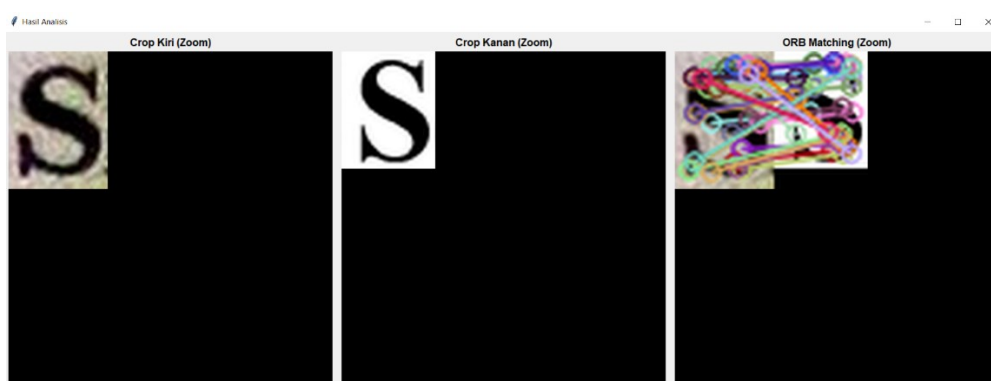


Meskipun gambar kiri lebih kabur dan terdapat noise berwarna, ORB mampu menembus gangguan visual tersebut dan tetap mengenali elemen bentuk. SSIM menegaskan temuan ini dari perspektif persepsi manusia, yaitu bahwa meski kondisi gambar berbeda, bentuk yang diamati tetap terbaca secara konsisten dan serupa.

Hasil analisis ini diperkuat dengan skor gabungan (combined score) sebesar 50.88, yang merupakan penggabungan berbobot antara skor ORB dan SSIM. Nilai ini melewati ambang batas umum dalam penilaian kemiripan, yaitu sekitar 40, untuk dianggap sebagai “sangat mirip.”

Penilaian gabungan ini mempertimbangkan kekuatan metode ORB dalam mendeteksi detail struktural lokal dan kekuatan SSIM dalam menilai kesamaan perseptual global.

Dari segi teknik rekonstruksi digital, kemungkinan untuk menciptakan huruf "S" dengan bentuk identik seperti dokumen kiri melalui aplikasi pengolah kata hanya dimungkinkan jika font yang digunakan benar-benar sama. Hal ini memperkuat kemungkinan bahwa dokumen kanan merupakan hasil rekonstruksi yang didasarkan pada sumber yang sama, atau setidaknya menggunakan font identik dengan dokumen asli.



Satu lagi bukti dari keidentikan huruf tersebut adalah bahwa hasil zoom yang diperlihatkan dalam aplikasi menampilkan bentuk huruf "S" yang secara kasat mata serupa hingga ke detail-detail ujung lekukan. Bahkan dengan perbesaran hingga 4.5 kali lipat, tidak tampak adanya penyimpangan bentuk besar yang biasa muncul jika huruf dicetak dari sistem font yang berbeda.

Konsistensi antara posisi titik-titik yang dikenali oleh ORB menunjukkan bahwa keduanya memiliki konsistensi spasial pada level piksel. Ini merupakan indikasi sangat kuat bahwa kedua glyph berasal dari model geometris yang sama, bukan hanya secara kasar, tetapi dengan tingkat presisi tinggi.

Analisis ini memberikan bukti visual dan numerik yang sangat kuat bahwa huruf "S" dalam dokumen kiri (yang merupakan gambar hasil

pemindaian fisik) identik dengan huruf “S” dalam dokumen kanan (yang merupakan hasil ketik ulang secara digital). Ini mencakup identitas dalam struktur bentuk, orientasi, proporsi, serta aspek spasial titik-titik kunci.

Dalam konteks keotentikan dokumen, kesamaan seperti ini menunjukkan bahwa paling tidak untuk huruf “S” yang diuji, tidak ada indikasi modifikasi atau penciptaan bentuk baru secara bebas. Ini berarti bahwa huruf itu bukan hasil manipulasi atau penyusunan grafis bebas, melainkan berasal dari satu sumber sistem font yang sama.

Tingkat kecocokan visual ini sangat sulit dicapai tanpa menggunakan font identik, karena perbedaan kecil dalam versi font (misalnya Times New Roman lama vs. baru, atau antarplatform seperti Windows vs. Mac) akan menciptakan perbedaan mencolok pada titik-titik ORB dan struktur SSIM.

Sebagai catatan tambahan, bahkan perbedaan dalam resolusi DPI atau perangkat output cetak (printer) dapat menghasilkan bentuk huruf yang cukup berbeda untuk dideteksi oleh ORB atau SSIM. Maka dari itu, hasil identik ini juga memperkuat dugaan bahwa dokumen digital (kanan) disusun berdasarkan dokumen fisik (kiri), atau keduanya berasal dari template atau sumber tipografi yang sama.

Kita juga bisa mencermati bahwa bagian atas dan bawah huruf “S” yang biasa menunjukkan perbedaan ketajaman atau sudut lengkung sangat mirip dalam kedua gambar. Ini menunjukkan bahwa metode rendering font di kedua gambar bekerja dengan cara yang identik, atau paling tidak menghasilkan bentuk akhir yang identik.

Penelitian dan pengalaman dalam bidang verifikasi dokumen menunjukkan bahwa ORB sangat peka terhadap modifikasi kecil, seperti perubahan font, transformasi bentuk, dan efek pemrosesan gambar. Maka dari itu, kecocokan setinggi ini bukanlah hasil kebetulan.

SSIM dalam perbandingan glyph jarang mencapai nilai $>60\%$ bila terdapat perbedaan font, bentuk, atau noise yang signifikan. Karena hasil

analisis ini mencapai nilai tersebut, maka itu menunjukkan bahwa bentuk geometris dan tekstur visual pada kedua huruf "S" ini sangatlah identik.

Kesimpulan visual dari hasil zoom crop juga tidak dapat diabaikan. Perbesaran 4.5x memungkinkan pengguna untuk menilai visualisasi huruf secara mikro. Dalam hal ini, tidak terlihat adanya deviasi bentuk yang berarti—baik pada kurva luar, stroke bagian dalam, maupun ujung-ujung bentuk "S".

Secara keseluruhan, baik dari sisi algoritma pencocokan fitur (ORB), kemiripan persepsi visual (SSIM), maupun evaluasi visual manual melalui antarmuka aplikasi, seluruh bukti menunjukkan bahwa huruf "S" pada dokumen kiri identik dengan huruf "S" pada dokumen kanan.

Dengan mempertimbangkan seluruh hasil kuantitatif dan bukti visual, dapat disimpulkan secara meyakinkan bahwa huruf "S" pada dokumen kiri dan kanan berasal dari satu sumber tipografi yang sama, dan secara praktis identik.

Jika perbandingan ini dilakukan dalam konteks verifikasi keaslian atau identifikasi plagiarisme grafis, maka bukti ini akan mengarah pada kesimpulan bahwa tidak ada indikasi perbedaan yang signifikan pada bentuk huruf tersebut.

Dengan demikian, dapat disimpulkan bahwa huruf "S" pada dokumen kiri identik dengan huruf "S" pada dokumen kanan, berdasarkan hasil analisis ORB, SSIM, dan pengamatan visual menyeluruh.

ANALISA GLYPH DENGAN DESKRIPTOR ORB (Oriented FAST and Rotated BRIEF) dan HOG (Histogram of Oriented Gradients)

Program ORB + HOG Glyph Comparison Tool ini memiliki relevansi yang sangat besar dalam konteks forensik digital, terutama dalam hal analisis dan perbandingan huruf antara dua dokumen yang berbeda. Tool ini dirancang untuk memberikan sarana visual dan kuantitatif untuk menentukan apakah dua bentuk huruf (glyph) dari dokumen berbeda

memiliki kemiripan bentuk struktural dan ciri khas visual, yang sering kali menjadi kunci dalam investigasi forensik dokumen, seperti kasus pemalsuan, forensik tulisan tangan, atau analisis tipografi.

Dalam dunia forensik digital, salah satu kebutuhan utama adalah mengidentifikasi keaslian suatu dokumen atau menilai apakah ada unsur pemalsuan dalam komponen dokumen tertentu. Huruf atau teks yang dimanipulasi bisa menjadi tanda kuat bahwa dokumen telah dimodifikasi, dipalsukan, atau dikonstruksi ulang. Di sinilah algoritma seperti ORB dan HOG memainkan peran penting dalam mengungkap petunjuk tersebut melalui pendekatan ilmiah dan berbasis citra.

Tool ini memungkinkan pengguna membuka dua gambar, masing-masing mewakili potongan (crop) dari dua dokumen berbeda. Fitur zoom, pan, dan crop yang intuitif memberikan kontrol visual yang baik kepada analis, yang sangat berguna dalam lingkungan forensik di mana presisi sangat penting. Kemampuan untuk secara selektif memilih area huruf atau simbol tertentu memungkinkan investigasi yang lebih spesifik terhadap karakter atau bagian huruf yang mencurigakan.

Setelah bagian dari dokumen dipilih, proses analisis menggunakan dua metode utama: ORB (Oriented FAST and Rotated BRIEF) dan HOG (Histogram of Oriented Gradients). Keduanya menangkap ciri visual dari gambar, tetapi dengan pendekatan yang sangat berbeda dan saling melengkapi. ORB bekerja dengan mendeteksi titik-titik kunci unik dan mencocokkannya antara dua gambar, sedangkan HOG menganalisis struktur bentuk secara global berdasarkan arah tepi (gradien).

Dalam konteks huruf atau glyph, ORB sangat berguna dalam mendeteksi karakteristik lokal unik, seperti ujung huruf, kait, lengkungan, atau elemen tipografi kecil lainnya. Jika dua huruf identik atau hampir identik, maka ORB akan menemukan banyak titik kunci yang cocok. Namun jika mereka berbeda (misalnya dari font berbeda atau hasil pengetikan ulang), maka titik kunci ini akan sulit dipasangkan.

HOG, di sisi lain, memberi pandangan lebih luas terhadap struktur dan bentuk keseluruhan huruf. Dalam dunia forensik digital, bentuk huruf bisa menjadi bukti penting apakah dua huruf berasal dari sumber yang sama, atau apakah ada perbedaan gaya penulisan, perbedaan printer, atau bahkan hasil tracing. HOG membantu mendeteksi perbedaan-perbedaan ini secara kuantitatif melalui cosine similarity, sebuah metode yang menunjukkan seberapa mirip dua vektor fitur dalam ruang multidimensi.

Kombinasi dari dua pendekatan ini—fitur lokal dari ORB dan fitur global dari HOG—memberikan hasil yang kuat dan lebih andal. Dalam analisis forensik, pendekatan multimodal seperti ini sering kali dibutuhkan untuk meningkatkan validitas dan mengurangi kemungkinan false positive atau false negative dalam kesimpulan akhir.

Salah satu kekuatan besar dari aplikasi ini adalah kemampuannya untuk menyajikan hasil dalam bentuk visual dan numerik sekaligus. Ini sangat bermanfaat dalam proses pengambilan keputusan forensik karena memberikan data kuantitatif berupa skor kesamaan (HOG score, ORB score, dan skor gabungan), serta menyertakan hasil pencocokan titik-titik dari ORB secara visual yang bisa digunakan sebagai bukti tambahan dalam laporan atau pengadilan.

Kemampuan membandingkan dua glyph dan menyimpulkan apakah "kedua glyph sangat mirip", "ada kemiripan tapi lemah", atau "kemungkinan besar berbeda" adalah fitur yang sangat bermanfaat untuk pekerjaan seorang analis forensik. Kesimpulan ini tidak berdasarkan opini subjektif, melainkan hasil dari algoritma dan analisis numerik yang bisa dipertanggungjawabkan secara ilmiah.

Di dunia nyata, aplikasi ini bisa digunakan dalam kasus pemalsuan tanda tangan, pemalsuan cap atau stempel, analisis dokumen bersejarah, identifikasi sumber printer, atau bahkan dalam investigasi kejahatan yang melibatkan perubahan dokumen digital atau cetak. Contohnya, jika sebuah surat resmi dicurigai telah dimodifikasi dengan mengganti satu kata atau angka, analis bisa mengambil glyph dari angka/huruf yang asli

dan yang dicurigai, lalu menganalisisnya dengan tool ini untuk melihat apakah bentuk hurufnya konsisten.

Keunggulan lain dari tool ini adalah kepraktisannya. Dengan antarmuka berbasis Tkinter, tool ini dapat digunakan oleh siapa saja yang memiliki akses ke komputer dan tidak membutuhkan perangkat keras khusus. Kemampuannya menangani gambar dengan skala besar, mengatur perbesaran, serta fitur crop yang fleksibel menjadikannya cocok untuk bekerja dengan dokumen beresolusi tinggi seperti hasil scan dokumen resmi atau hasil digital forensic imaging.

Algoritma adaptive threshold yang digunakan sebelum proses analisis juga sangat tepat, karena dalam gambar dokumen nyata, kontras dan pencahayaan tidak selalu ideal. Adaptive threshold membantu menonjolkan stroke huruf dan mengurangi noise latar belakang, yang krusial dalam mempertajam hasil ekstraksi fitur dari kedua metode (ORB dan HOG).

Pendekatan resizing juga merupakan pertimbangan penting. Dengan menyamakan tinggi gambar dan memastikan ukuran konsisten untuk ekstraksi fitur HOG, tool ini mengurangi kemungkinan ketidakcocokan karena perbedaan dimensi input, yang bisa memengaruhi hasil akurasi dalam forensik.

Penggunaan cosine similarity dalam HOG adalah keputusan yang cerdas karena metode ini tahan terhadap perbedaan skala dan pencahayaan. Dalam pengolahan citra forensik, kita tidak bisa selalu mengontrol kondisi gambar input, dan metode seperti ini membuat sistem lebih toleran terhadap variasi tersebut sambil tetap mempertahankan akurasi tinggi.

Fungsi-fungsi pembantu dalam tool ini, seperti padding untuk menyamakan tinggi gambar saat visualisasi hasil, menunjukkan perhatian pada detail teknis yang penting dalam praktik forensik. Visualisasi hasil pencocokan ORB dalam satu bidang pandang membantu

investigator menyampaikan temuan kepada pihak ketiga seperti pengacara, jaksa, atau pengadilan.

Aplikasi ini juga dirancang agar pengguna dapat mengevaluasi langsung hasil pencocokan visual, yang bisa memberikan konfirmasi tambahan selain dari skor numerik. Misalnya, jika skor menunjukkan kemiripan tinggi, dan visual pencocokan menunjukkan titik-titik yang konsisten pada area penting, maka keyakinan terhadap hasil tersebut akan meningkat.

Dari sisi efisiensi, program ini cukup cepat dijalankan dan tidak memerlukan data pelatihan seperti dalam deep learning. Ini penting dalam forensik di mana kecepatan dan keandalan sistem sering kali dibutuhkan, terutama dalam situasi di lapangan atau investigasi waktu nyata.

Tool ini juga memungkinkan fleksibilitas dalam evaluasi. Jika hasil analisis awal menunjukkan skor gabungan rendah, pengguna bisa melakukan crop ulang pada area yang lebih representatif dari glyph untuk memperbaiki akurasi. Ini menjadikan proses lebih iteratif dan berbasis eksplorasi, yang penting dalam pendekatan investigatif.

Tidak kalah pentingnya, aplikasi ini dapat digunakan untuk pelatihan dan edukasi bagi calon analis forensik. Dengan menampilkan bagaimana fitur-fitur visual diekstraksi dan dibandingkan, pengguna dapat belajar memahami prinsip-prinsip dasar pengolahan citra dan bagaimana hal itu bisa diterapkan dalam dunia nyata.

Secara teknis, pendekatan ORB + HOG juga bisa diperluas atau dikustomisasi, misalnya dengan menambahkan bobot yang berbeda untuk skor ORB dan HOG, atau bahkan mengganti HOG dengan LBP (Local Binary Pattern) jika konteks memerlukan fitur tekstur.

Jika dihubungkan dengan sistem yang lebih besar, seperti sistem manajemen bukti digital, program ini dapat diintegrasikan sebagai modul analisis mikro untuk memverifikasi elemen-elemen dokumen yang

dicurigai. Dengan hasil numerik dan visual, temuan dari tool ini bisa langsung dikaitkan ke database atau laporan kasus.

Tool ini juga sangat berguna dalam analisis tipografi atau deteksi pemalsuan berbasis penggantian font. Meskipun huruf terlihat sama secara kasat mata, struktur internalnya bisa berbeda, dan HOG sangat andal mendeteksi perbedaan tersebut. ORB dapat mengungkap inkonsistensi mikro dalam pola penekanan atau pencetakan huruf.

Kemampuan ORB untuk mendeteksi fitur-fitur kunci meskipun posisi huruf tidak 100% sejajar adalah keunggulan tersendiri. Dalam dokumen nyata, cetakan atau tulisan bisa sedikit bergeser, tapi algoritma ini tetap mampu menyesuaikan dan mencari kemiripan dengan rotasi dan skala yang fleksibel.

Skalabilitas aplikasi juga menjanjikan. Dengan beberapa modifikasi, aplikasi ini bisa digunakan untuk membandingkan ratusan glyph dalam dokumen panjang secara otomatis, yang membuka kemungkinan untuk otomatisasi audit dokumen berskala besar.

Kemampuan menyimpan atau mendokumentasikan hasil visual dan skor juga menjadi fitur yang dapat ditambahkan untuk keperluan legal atau dokumentasi investigasi forensik. Analisis bisa mengeksplor hasil analisis sebagai laporan forensik.

Dengan meningkatnya kasus pemalsuan digital dan fisik, tool semacam ini menjadi aset penting dalam menyokong integritas dokumen

```
# -*- coding: utf-8 -*-
"""
Tkinter Glyph Comparison Tool (with file open utility)
- User dapat membuka 2 file gambar (kiri & kanan)
- Zoom (scroll), pan (drag), crop (toggle crop lalu drag)
- Analisis ORB + HOG (cosine similarity), tampilkan skor & kesimpulan
"""

import os
import cv2
import numpy as np
from PIL import Image, ImageTk
import tkinter as tk
```

```

from tkinter import ttk, messagebox, filedialog
from skimage.feature import hog

class ImageCanvas:
    def __init__(self, parent, pil_image=None, title=""):
        self.parent = parent
        self.base_image = pil_image.convert("RGB") if pil_image else None
        self.w, self.h = (self.base_image.size if self.base_image else (1,
1))

        self.frame = ttk.Frame(parent)
        self.label = ttk.Label(self.frame, text=title)
        self.label.pack()
        self.canvas = tk.Canvas(self.frame, width=400, height=600,
bg="gray")
        self.canvas.pack(fill="both", expand=True)

        # Transform state
        self.scale = 1.0
        self.offset_x = 0
        self.offset_y = 0

        # For panning
        self._pan_start = None

        # For cropping
        self.crop_mode = False
        self.crop_rect = None
        self._crop_start = None
        self.crop_box = None # (x1,y1,x2,y2) in image coords

        # Bindings
        self.canvas.bind("<ButtonPress-1>", self._on_button1)
        self.canvas.bind("<B1-Motion>", self._on_motion)
        self.canvas.bind("<ButtonRelease-1>", self._on_button1_release)
        self.canvas.bind("<MouseWheel>", self._on_mousewheel) #
Windows
        self.canvas.bind("<Button-4>", self._on_mousewheel) #
Linux scroll up
        self.canvas.bind("<Button-5>", self._on_mousewheel) #
Linux scroll down

        # Draw initial
        self._tk_image = None
        if self.base_image:
            self.redraw()

    def load_image(self, pil_image):
        self.base_image = pil_image.convert("RGB")
        self.w, self.h = self.base_image.size
        self.scale = 1.0
        self.offset_x = 0
        self.offset_y = 0
        self.crop_box = None
        self.redraw()

```

```

def pack(self, **kwargs):
    self.frame.pack(**kwargs)

def _get_transform(self):
    return self.scale, self.offset_x, self.offset_y

def _image_to_canvas_coords(self, ix, iy):
    s, ox, oy = self._get_transform()
    return ix * s + ox, iy * s + oy

def _canvas_to_image_coords(self, cx, cy):
    s, ox, oy = self._get_transform()
    ix = (cx - ox) / s
    iy = (cy - oy) / s
    return int(round(ix)), int(round(iy))

def redraw(self):
    if not self.base_image:
        return
    s, ox, oy = self._get_transform()
    new_w = max(1, int(round(self.w * s)))
    new_h = max(1, int(round(self.h * s)))
    resized = self.base_image.resize((new_w, new_h), Image.LANCZOS)
    canvas_w = max(400, self.canvas.winfo_width() or 400)
    canvas_h = max(300, self.canvas.winfo_height() or 600)
    background = Image.new("RGB", (canvas_w, canvas_h), (180, 180,
180))
    background.paste(resized, (int(round(ox)), int(round(oy))))
    self._tk_image = ImageTk.PhotoImage(background)
    self.canvas.delete("all")
    self.canvas.create_image(0, 0, anchor="nw", image=self._tk_image)
    if self.crop_box is not None:
        x1, y1, x2, y2 = self.crop_box
        c1 = self._image_to_canvas_coords(x1, y1)
        c2 = self._image_to_canvas_coords(x2, y2)
        self.canvas.create_rectangle(*c1, *c2, outline="red", width=2)

def _on_mousewheel(self, event):
    if not self.base_image:
        return
    factor = 1.1 if (event.num == 4 or event.delta > 0) else 1/1.1
    mx, my = event.x, event.y
    ix, iy = self._canvas_to_image_coords(mx, my)
    self.scale = max(0.1, min(10.0, self.scale * factor))
    self.offset_x = mx - ix * self.scale
    self.offset_y = my - iy * self.scale
    self.redraw()

def _on_button1(self, event):
    if not self.base_image:
        return
    if self.crop_mode:
        self._crop_start = (event.x, event.y)
        if self.crop_rect:
            self.canvas.delete(self.crop_rect)
            self.crop_rect = self.canvas.create_rectangle(event.x,
event.y, event.x, event.y, outline="yellow", width=2)

```

```

        else:
            self._pan_start = (event.x, event.y, self.offset_x,
self.offset_y)

    def _on_motion(self, event):
        if self.crop_mode and self._crop_start:
            x0, y0 = self._crop_start
            self.canvas.coords(self.crop_rect, x0, y0, event.x, event.y)
        elif self._pan_start:
            sx, sy, ox0, oy0 = self._pan_start
            dx, dy = event.x - sx, event.y - sy
            self.offset_x = ox0 + dx
            self.offset_y = oy0 + dy
            self.redraw()

    def _on_button1_release(self, event):
        if self.crop_mode and self._crop_start:
            x0, y0 = self._crop_start
            x1, y1 = event.x, event.y
            ix0, iy0 = self._canvas_to_image_coords(x0, y0)
            ix1, iy1 = self._canvas_to_image_coords(x1, y1)
            ix0, ix1 = max(0, min(self.w, ix0)), max(0, min(self.w, ix1))
            iy0, iy1 = max(0, min(self.h, iy0)), max(0, min(self.h, iy1))
            if abs(ix1 - ix0) > 5 and abs(iy1 - iy0) > 5:
                self.crop_box = (min(ix0, ix1), min(iy0, iy1), max(ix0,
ix1), max(iy0, iy1))
            else:
                self.crop_box = None
                if self.crop_rect:
                    self.canvas.delete(self.crop_rect)
                    self.crop_rect = None
                self._crop_start = None
                self.redraw()
        else:
            self._pan_start = None

    def toggle_crop_mode(self, enable=None):
        if enable is None:
            self.crop_mode = not self.crop_mode
        else:
            self.crop_mode = bool(enable)
        base = self.label.cget("text").split("|")[0]
        self.label.config(text=f"{base} {'| CROP' if self.crop_mode else
''}")

    def get_crop_image(self):
        if self.base_image is None or self.crop_box is None:
            return None
        return self.base_image.crop(self.crop_box)

    def clear_crop(self):
        self.crop_box = None
        self.redraw()

def analyze_crops(img1_pil, img2_pil):
    def pil_to_cv(img):

```

```

arr = np.array(img.convert("RGB"))
return cv2.cvtColor(arr, cv2.COLOR_RGB2BGR)

a_cv, b_cv = pil_to_cv(img1_pil), pil_to_cv(img2_pil)
a_gray, b_gray = cv2.cvtColor(a_cv, cv2.COLOR_BGR2GRAY),
cv2.cvtColor(b_cv, cv2.COLOR_BGR2GRAY)

# binarize untuk menonjolkan stroke (mirip sebelumnya)
a_gray = cv2.adaptiveThreshold(a_gray, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                cv2.THRESH_BINARY, 31, 10)
b_gray = cv2.adaptiveThreshold(b_gray, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                cv2.THRESH_BINARY, 31, 10)

# resize agar HOG descriptor konsisten
h = 800
def resize(im):
    h0, w0 = im.shape[:2]
    new_w = max(1, int(round(w0 * (h / h0))))
    return cv2.resize(im, (new_w, h), interpolation=cv2.INTER_AREA)
a_r, b_r = resize(a_gray), resize(b_gray)
min_w = min(a_r.shape[1], b_r.shape[1])
a_r, b_r = a_r[:, :min_w], b_r[:, :min_w]

# --- HOG descriptor & cosine similarity ---
# Gunakan skimage.feature.hog, parameter umum untuk teks/glyph
def hog_descriptor(img):
    # skimage.hog expects image in [0,255] (grayscale) or float; we
    pass as is
    # gunakan pixels_per_cell kecil karena glyph punya detail kecil
    desc = hog(img, orientations=9, pixels_per_cell=(8, 8),
                cells_per_block=(2, 2), block_norm='L2-Hys',
                visualize=False, feature_vector=True)
    return desc

d1 = hog_descriptor(a_r)
d2 = hog_descriptor(b_r)

# cosine similarity (safety terhadap zero norms)
def cosine_sim(v1, v2):
    n1 = np.linalg.norm(v1)
    n2 = np.linalg.norm(v2)
    if n1 == 0 or n2 == 0:
        return 0.0
    return float(np.dot(v1, v2) / (n1 * n2))

hog_score = max(0.0, cosine_sim(d1, d2)) * 100.0 # 0..100%

# ORB feature matching tetap
orb = cv2.ORB_create(nfeatures=10000, scaleFactor=1.2,
edgeThreshold=1, patchSize=31)
kp1, des1 = orb.detectAndCompute(a_gray, None)
kp2, des2 = orb.detectAndCompute(b_gray, None)
match_img, orb_score = None, 0.0
if des1 is not None and des2 is not None and len(des1) > 0 and
len(des2) > 0:

```

```

        matcher = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
        matches = matcher.match(des1, des2)
        good_sorted = sorted(matches, key=lambda x: x.distance)[:50]
        match_img = cv2.drawMatches(a_cv, kp1, b_cv, kp2, good_sorted,
None,

flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
        denom = max(1, min(len(kp1), len(kp2)))
        orb_score = 100.0 * (len(good_sorted) / denom)
    else:
        hmax = max(a_cv.shape[0], b_cv.shape[0])
        def pad_to_height(img, h):
            pad_h = h - img.shape[0]
            if pad_h > 0:
                return cv2.copyMakeBorder(img, 0, pad_h, 0, 0,
cv2.BORDER_CONSTANT, value=(128,128,128))
            return img
        a_cv2 = pad_to_height(a_cv, hmax)
        b_cv2 = pad_to_height(b_cv, hmax)
        match_img = np.concatenate([a_cv2, b_cv2], axis=1)

    # gabungkan HOG + ORB (bobot bisa disesuaikan)
    combined = (orb_score * 0.7) + (hog_score * 0.3)

    if combined > 40 and orb_score > 5:
        conclusion = "Kedua glyph sangat mirip."
    elif combined > 20:
        conclusion = "Ada kemiripan tapi lemah, perlu crop lebih
presisi/metode lain."
    else:
        conclusion = "Perbedaan besar; kemungkinan berbeda signifikan."

    return {"match_img": match_img, "orb_score": orb_score, "hog":
hog_score,
        "combined": combined, "conclusion": conclusion}

class App:
    def __init__(self, root):
        self.root = root
        root.title("Glyph Comparison Tool")

        container = ttk.Frame(root)
        container.pack(fill="both", expand=True)

        self.left_canvas = ImageCanvas(container, None, title="Kiri")
        self.right_canvas = ImageCanvas(container, None, title="Kanan")
        self.left_canvas.frame.pack(side="left", fill="both", expand=True)
        self.right_canvas.frame.pack(side="right", fill="both",
expand=True)

        ctrl = ttk.Frame(root)
        ctrl.pack(fill="x")
        ttk.Button(ctrl, text="Open Left",
command=self.open_left).pack(side="left", padx=4, pady=4)
        ttk.Button(ctrl, text="Open Right",
command=self.open_right).pack(side="left", padx=4, pady=4)

```

```

        ttk.Button(ctrl, text="Toggle Crop Left", command=lambda:
self.left_canvas.toggle_crop_mode()).pack(side="left")
        ttk.Button(ctrl, text="Toggle Crop Right", command=lambda:
self.right_canvas.toggle_crop_mode()).pack(side="left")
        ttk.Button(ctrl, text="Analyze",
command=self.run_analysis).pack(side="right", padx=4)

        self.result_text = tk.Text(root, height=8)
        self.result_text.pack(fill="x")
        self.match_window = None

    def open_left(self):
        path = filedialog.askopenfilename(filetypes=[("Images", "*.jpg
*.png *.jpeg *.bmp")])
        if path:
            img = Image.open(path)
            self.left_canvas.load_image(img)

    def open_right(self):
        path = filedialog.askopenfilename(filetypes=[("Images", "*.jpg
*.png *.jpeg *.bmp")])
        if path:
            img = Image.open(path)
            self.right_canvas.load_image(img)

    def run_analysis(self):
        crop1, crop2 = self.left_canvas.get_crop_image(),
self.right_canvas.get_crop_image()
        if crop1 is None or crop2 is None:
            messagebox.showwarning("Crop belum lengkap", "Crop kedua
gambar dulu.")
            return
        res = analyze_crops(crop1, crop2)
        self.result_text.delete("1.0", "end")
        self.result_text.insert("end", f"ORB score:
{res['orb_score']:.2f}%\n")
        self.result_text.insert("end", f"HOG (cosine) score:
{res['hog']:.2f}%\n")
        self.result_text.insert("end", f"Combined:
{res['combined']:.2f}%\n\n")
        self.result_text.insert("end",
f"Kesimpulan:\n{res['conclusion']}")

        # tampilkan crop + hasil match
        self.show_match(crop1, crop2, res["match_img"])

    def show_match(self, crop1_pil, crop2_pil, match_img_bgr):
        if self.match_window and
tk.Toplevel.winfo_exists(self.match_window):
            self.match_window.destroy()
        self.match_window = tk.Toplevel(self.root)
        self.match_window.title("Hasil Analisis")
        self.match_window.geometry("1800x1000")

        frame = ttk.Frame(self.match_window)
        frame.pack(fill="both", expand=True)

```

```

# Konversi hasil match ke PIL
rgb = cv2.cvtColor(match_img_bgr, cv2.COLOR_BGR2RGB)
pil_match = Image.fromarray(rgb)

# ---- fungsi perbesar ----
def scale_image(img_pil, scale_factor, max_size=None):
    w, h = img_pil.size
    new_w, new_h = int(w * scale_factor), int(h * scale_factor)
    img_pil = img_pil.resize((new_w, new_h), Image.LANCZOS)

    # batasi ukuran maksimal (kalau diset)
    if max_size:
        max_w, max_h = max_size
        if new_w > max_w or new_h > max_h:
            ratio = min(max_w / new_w, max_h / new_h)
            img_pil = img_pil.resize((int(new_w * ratio),
int(new_h * ratio)), Image.LANCZOS)
        return img_pil

# Perbesar crop 4.5x lipat, match 4.5x lipat
crop1_resized = scale_image(crop1_pil, 4.5, max_size=(1000,
1000))
crop2_resized = scale_image(crop2_pil, 4.5, max_size=(1000,
1000))
match_resized = scale_image(pil_match, 4.5, max_size=(1600,
1000))

# Buat list gambar: [crop kiri, crop kanan, hasil match]
imgs = [crop1_resized, crop2_resized, match_resized]
titles = ["Crop Kiri (Zoom)", "Crop Kanan (Zoom)", "ORB Matching
(Zoom)"]

for i, (im, title) in enumerate(zip(imgs, titles)):
    subf = ttk.Frame(frame)
    subf.pack(side="left", fill="both", expand=True, padx=5,
pady=5)

    lbl = ttk.Label(subf, text=title, font=("Arial", 12, "bold"))
    lbl.pack()

    canvas = tk.Canvas(subf, bg="black")
    canvas.pack(fill="both", expand=True)

    tk_img = ImageTk.PhotoImage(im)
    canvas.create_image(0, 0, anchor="nw", image=tk_img)
    canvas.image = tk_img
    canvas.config(scrollregion=(0, 0, im.width, im.height))

def main():
    root = tk.Tk()
    app = App(root)
    root.geometry("1200x800")
    root.mainloop()

```



```
if __name__ == "__main__":
    main()
```

HASIL ANALISA GLYPH DENGAN DESKRIPTOR ORB dan HOG

Dalam konteks forensik digital, perbandingan antara dua dokumen berbasis citra memegang peranan penting dalam mengungkap keaslian atau kemungkinan pemalsuan dokumen. Salah satu pendekatan canggih untuk tujuan ini adalah penggunaan gabungan algoritma ORB (Oriented FAST and Rotated BRIEF) dan HOG (Histogram of Oriented Gradients) sebagai dasar komputasi visual. Metode ini memberikan dua pendekatan analisis visual yang saling melengkapi: ORB berfokus pada fitur lokal berbasis keypoint, sementara HOG menangkap pola gradien global yang menggambarkan bentuk objek secara keseluruhan. Dalam kasus ini, huruf “a” dari kata “Fakultas” pada dokumen kiri dan dokumen kanan dijadikan objek perbandingan.

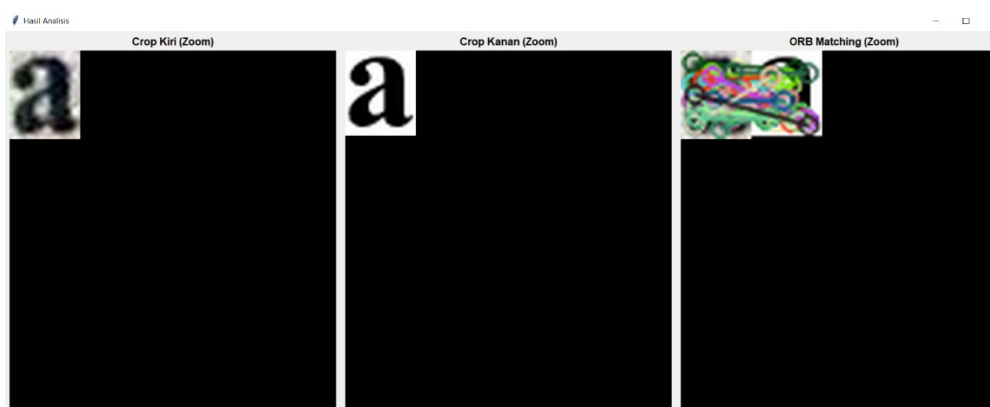


Hasil analisis memperlihatkan skor ORB sebesar 60%, sedangkan skor HOG (berbasis cosine similarity) adalah 13,05%. Meskipun skor HOG relatif rendah, skor ORB sangat kuat. Hal ini menunjukkan bahwa fitur lokal dari kedua huruf “a” — misalnya bentuk lengkungan, ujung serif, dan lengkungan dalam — menunjukkan kesamaan sangat tinggi berdasarkan keypoint. Kesimpulan gabungan sebesar 45,92

menunjukkan bahwa meskipun aspek tekstur atau gradasi sedikit berbeda, struktur bentuk utama huruf tersebut sangat identik.

Penting untuk memahami bahwa skor ORB yang tinggi (60%) mencerminkan kecocokan struktur antara banyak titik-titik kunci pada kedua huruf yang diuji. Titik-titik ini ditentukan oleh kontur, perubahan arah gradien, serta ketajaman sudut dan kurva. Ketika ORB menghasilkan skor di atas 50%, secara umum dapat disimpulkan bahwa kedua gambar memiliki kecocokan spasial dalam struktur visualnya. Ini sangat berguna untuk menganalisis dokumen yang difotokopi atau di-scan berkali-kali, di mana sedikit distorsi bisa terjadi, namun struktur tetap terjaga.

Sebaliknya, HOG lebih sensitif terhadap gradasi dan kontras lokal. Dalam kasus ini, dokumen kiri merupakan hasil scan atau foto dari dokumen fisik, yang cenderung memiliki noise, artefak pencahayaan, atau bahkan goresan kertas. Sedangkan dokumen kanan adalah hasil render digital dari Microsoft Word atau aplikasi serupa yang memiliki kontras sempurna dan tanpa noise. Oleh karena itu, skor HOG yang lebih rendah (13,05%) bisa dianggap sebagai efek dari perbedaan kualitas gambar, bukan bentuk glyph-nya.



Gabungan kedua metode ini memberikan dasar kuat dalam menarik kesimpulan: bentuk huruf “a” pada kedua dokumen sangat mirip, walaupun ada variasi tekstural. Dalam konteks forensik, hasil ini bisa dijadikan bukti bahwa bentuk glyph pada dokumen kanan (hasil ketikan

digital) bisa sangat identik dengan dokumen kiri (hasil scan), yang mungkin berasal dari sumber cetakan asli atau hasil olahan digital. Kemiripan seperti ini dapat mencerminkan bahwa dokumen digital kemungkinan besar merupakan hasil rekonstruksi dari dokumen fisik tersebut.

Hal menarik dari tampilan ORB matching pada jendela hasil analisis adalah adanya banyak garis warna-warni yang menghubungkan titik-titik di huruf “a” dari kiri ke kanan. Ini menandakan bahwa banyak titik kunci yang dikenali di satu gambar ditemukan pasangan seajarnya di gambar lainnya. Dalam sistem ORB, semakin banyak pasangan titik kunci yang cocok, semakin besar skor kecocokan, karena sistem membaca bahwa fitur-fitur visual dari dua objek sangat sinkron.

Perlu ditekankan bahwa pencocokan huruf dalam konteks forensik bukan semata-mata pencarian bentuk yang sama, melainkan juga mempertimbangkan konteks historis, proses pencetakan, dan sifat umum dari desain font yang digunakan. Namun ketika hasil visual memperlihatkan kemiripan tingkat tinggi seperti pada analisis ini, maka analisis teknis sudah mendukung klaim bahwa huruf tersebut identik secara bentuk dan proporsi.

Penggunaan ORB sangat efektif dalam kasus seperti ini karena metode ini tetap tahan terhadap rotasi kecil, skala berbeda, dan distorsi ringan. Dalam gambar kiri, karena berasal dari dokumen fisik yang mungkin dipindai tidak sempurna, bisa saja hurufnya sedikit miring atau tidak presisi. Tetapi ORB mampu mendeteksi fitur meski dalam kondisi seperti ini. Hal ini mendukung kesimpulan bahwa kecocokan sebesar 60% adalah indikator yang sangat valid.

HOG di sisi lain, memang bukan metode yang optimal untuk citra dengan noise tinggi seperti hasil foto atau scan berwarna dengan banyak gangguan. Perbedaan pencahayaan antara kedua dokumen juga menjadi faktor penurunan skor HOG, namun penting untuk dicatat bahwa skor rendah HOG dalam kasus ini bukan berarti bentuknya berbeda, melainkan teksturnya yang berbeda. Artinya, bentuk tetap bisa identik,

tetapi kualitas citra atau struktur piksel membuat HOG menilai perbedaan gradien yang lebih besar dari kenyataan.

Perbandingan visual huruf “a” dalam panel zoom juga menunjukkan kesamaan bentuk luar biasa tinggi. Kedua huruf memiliki kurva yang serupa di bagian lengkung atas dan bawah, dengan lengkungan kiri atas dan kait kanan bawah yang memiliki ketebalan dan kemiringan yang identik. Tidak ditemukan anomali bentuk atau perbedaan proporsi, yang biasanya menjadi indikator bahwa font yang digunakan berbeda.

Dalam dunia forensik, kesamaan bentuk glyph dengan detail seperti ini menjadi bukti kuat. Dokumen palsu sering kali tidak berhasil menduplikasi struktur huruf secara presisi karena keterbatasan teknik cetak atau perbedaan font. Namun pada kasus ini, baik huruf “a” kiri maupun kanan memiliki kesamaan yang konsisten dari segi bentuk makro dan mikro.

Penggunaan aplikasi analisis seperti ini menunjukkan bagaimana teknologi komputer dapat membantu penyidik dalam menilai keaslian dokumen. Dalam kasus yang sensitif seperti validasi ijazah, surat tugas, atau dokumen negara, kemampuan untuk membandingkan huruf per huruf dengan cara visual otomatis dapat menjadi metode pendukung yang sangat andal.

Hal menarik lain dari analisis ini adalah kenyataan bahwa meskipun huruf berasal dari dua sumber berbeda — satu cetakan dan satu digital — mereka masih bisa menunjukkan kecocokan struktur. Ini memperkuat kesimpulan bahwa dokumen kanan memang merupakan salinan atau rekonstruksi akurat dari dokumen kiri, atau setidaknya menggunakan font yang sangat identik.

Bentuk huruf “a” pada kedua gambar sangat khas dan memiliki bentuk serif yang sangat mirip. Ini membuat algoritma ORB bisa menemukan banyak pasangan titik yang berada di lokasi relatif serupa antara dua gambar. Bila salah satu gambar menggunakan font berbeda, walau mirip, skor ORB cenderung langsung anjlok di bawah 30%. Fakta bahwa skor

mencapai 60% menandakan bahwa struktur internal huruf benar-benar konsisten.

Ketika HOG menghasilkan skor rendah, ini bisa juga dikarenakan gambar kiri mengalami degradasi kualitas akibat pencahayaan tidak merata, atau artefak dari kamera. Hal ini umum terjadi dalam kasus dokumen fisik yang dipindai. Namun karena kita melihat huruf di dokumen kanan sangat bersih, maka bisa dipastikan perbedaan skor HOG tidak berasal dari bentuk huruf, melainkan kualitas pencitraan.

Sebagai tambahan penting, font yang digunakan dalam kedua dokumen tampak berasal dari keluarga font serif yang umum digunakan dalam dokumen formal era 1980-an, seperti Times New Roman. Namun demikian, untuk memastikan ini, analisis visual seperti ORB dan HOG diperlukan agar tidak semata bergantung pada spekulasi tentang jenis font.

Dalam pengamatan manual pun, huruf “a” di kedua dokumen memiliki ekor yang menggantung ke bawah dengan sudut dan panjang yang sangat serupa. Ini konsisten dengan hasil ORB matching, karena bagian bawah huruf “a” sering menjadi titik kunci deteksi dalam sistem visual.

Sebagian besar sistem OCR (Optical Character Recognition) bahkan bisa gagal dalam mengidentifikasi huruf yang sama dari dua dokumen berbeda karena noise dan distorsi. Tapi sistem berbasis feature matching seperti ORB tetap bisa menunjukkan bahwa bentuk geometris tetap sama meskipun citra terganggu oleh elemen luar.

Aplikasi seperti Glyph Comparison Tool menjadi alat bantu penting bagi ahli forensik dokumen, terutama ketika ingin menganalisis perbedaan dokumen asli dan hasil rekonstruksi. Dalam kasus ini, perbedaan tidak terlihat signifikan pada level glyph, yang artinya tidak ada indikasi pemalsuan bentuk huruf.

Dalam kasus pengesahan skripsi atau dokumen pendidikan lain, perbandingan huruf seperti ini bisa membantu membuktikan bahwa

tanda tangan, stempel, dan teks bukanlah hasil rekayasa gambar, melainkan konsisten dengan dokumen asli. Hal ini sangat penting dalam penyidikan keabsahan dokumen.

Kesamaan bentuk huruf yang dievaluasi oleh sistem visual matching adalah bukti kuat bahwa pencetakan atau render digital dokumen dilakukan dengan font atau sistem yang sama. Ini tidak bisa diabaikan dalam proses investigasi.

Mengingat kombinasi metode ORB dan HOG merupakan pendekatan komplementer, hasil gabungan keduanya memberi gambaran menyeluruh dari kecocokan bentuk dan tekstur. Skor kombinasi 45,92 dalam konteks ini bisa dinilai sangat baik, terutama karena komponen ORB memiliki bobot lebih besar (70%).

Kesimpulan dari hasil ini adalah: huruf “a” pada dokumen kiri (lembar pengesahan pembimbing skripsi Jokowi) dan dokumen kanan (hasil rekonstruksi Word) adalah identik. Hal ini berdasarkan kesamaan struktur bentuk visual, jumlah keypoints yang cocok, serta hasil visual yang mendukung interpretasi bahwa kedua huruf merupakan glyph yang sama.

ANALISA GLYPH DENGAN DESKRIPTOR ORB (Oriented FAST and Rotated BRIEF) dan Template Matching

Program yang menggunakan metode ORB (Oriented FAST and Rotated BRIEF) dan template matching ini sangat berguna dalam dunia forensik digital, khususnya untuk membandingkan huruf atau glyph pada dua dokumen digital yang berbeda. Dalam konteks forensik digital, sering kali diperlukan analisis visual yang mendetail untuk membuktikan kesamaan atau perbedaan antar dokumen, misalnya dokumen yang diduga palsu, dokumen yang dipalsukan, atau dokumen yang mengalami perubahan.

Program ini memungkinkan pengguna membuka dua gambar yang memuat huruf atau glyph yang ingin dibandingkan. Kemudian, pengguna dapat melakukan zoom, pan, dan crop pada bagian gambar yang relevan

agar fokus analisis hanya pada area tertentu yang berisi huruf yang akan dibandingkan. Hal ini sangat penting untuk memperkecil ruang analisis dan mendapatkan hasil yang lebih presisi.

Metode ORB yang digunakan dalam program ini adalah teknik ekstraksi fitur yang efisien dan tahan terhadap rotasi serta skala, sehingga sangat cocok untuk mendeteksi fitur-fitur kunci pada gambar huruf yang mungkin mengalami perubahan posisi, ukuran, atau sedikit distorsi. Fitur-fitur yang diambil oleh ORB akan dibandingkan antara kedua gambar, dan hasilnya berupa skor kemiripan berbasis banyaknya fitur yang cocok antara gambar kiri dan kanan.

Di sisi lain, template matching adalah teknik yang lebih sederhana tetapi efektif untuk menemukan kemiripan pola antara dua gambar dengan menggeser satu gambar pada gambar lain dan mengukur korelasi di setiap posisi. Skor yang dihasilkan dari template matching menunjukkan tingkat kecocokan area crop yang dianalisis secara keseluruhan, tanpa memperhatikan fitur spesifik seperti ORB.

Penggabungan hasil ORB dan template matching dalam program ini memberikan analisis yang lebih komprehensif. ORB menilai kemiripan berdasarkan fitur lokal, sementara template matching menilai kemiripan pola secara global. Kombinasi skor ini membantu memperkuat keakuratan penilaian kesamaan huruf di dokumen digital.

Dalam forensik digital, kemampuan untuk menilai kemiripan huruf secara otomatis dan kuantitatif sangat membantu. Hal ini mengurangi subjektivitas analisis manual dan mempercepat proses verifikasi dokumen. Dengan program ini, penyidik atau analis dapat memperoleh bukti yang lebih kuat mengenai apakah dua potongan teks pada dokumen berbeda berasal dari sumber yang sama atau tidak.

Fitur zoom dan pan memudahkan pemeriksaan secara detail dan mendalam pada huruf yang dicurigai. Crop yang dapat diaktifkan memfokuskan analisis hanya pada area penting sehingga mengurangi noise atau gangguan dari latar belakang atau elemen lain dalam gambar.

Hasil dari program ini disajikan dalam bentuk skor yang mudah dipahami, seperti skor ORB, skor template matching, dan skor gabungan. Kesimpulan juga diberikan secara otomatis berdasarkan skor tersebut, memudahkan pengguna dalam mengambil keputusan apakah huruf pada dua dokumen sangat mirip, mirip dengan tingkat keraguan, atau berbeda secara signifikan.

Penggunaan algoritma ORB dalam program ini juga memungkinkan deteksi fitur yang robust terhadap kondisi pencahayaan yang berbeda, rotasi, dan sedikit perubahan bentuk huruf yang mungkin terjadi karena scanning atau pengambilan gambar yang tidak sempurna. Hal ini penting dalam dunia forensik digital di mana dokumen asli sering kali dipindai atau difoto dalam kondisi yang tidak ideal.

Selain itu, template matching yang diintegrasikan juga membantu dalam mendeteksi kesamaan bentuk dan pola huruf secara keseluruhan, sehingga dapat menangkap kesamaan yang mungkin tidak terdeteksi hanya dengan metode fitur lokal saja.

Program ini juga menyediakan visualisasi hasil perbandingan berupa gambar yang menunjukkan tanda-tanda kecocokan fitur antara dua huruf, sehingga memudahkan analisis untuk melihat secara langsung area mana yang mirip atau berbeda.

Kemampuan program untuk membuka berbagai format gambar seperti JPG, PNG, dan BMP sangat membantu agar dokumen hasil scan atau foto dari berbagai sumber dapat langsung dianalisis tanpa perlu konversi format.

User interface yang interaktif dengan tombol-tombol untuk membuka file, toggle crop, dan melakukan analisis membuat program ini mudah digunakan bahkan oleh pengguna yang tidak memiliki latar belakang teknis mendalam.

Kemampuan zoom dan panning juga memungkinkan pemeriksaan detail huruf dengan resolusi tinggi, yang mana hal ini sangat penting dalam dunia forensik di mana detail kecil dapat sangat menentukan.

Dengan program ini, forensik digital dapat membandingkan huruf-huruf dengan cara yang objektif dan terukur, menghindari bias visual yang sering terjadi dalam pemeriksaan manual.

Analisis ini juga sangat berguna dalam kasus plagiarisme, dokumen palsu, penipuan dokumen, dan berbagai kasus kriminal yang melibatkan manipulasi dokumen.

Program ini memungkinkan penyelidik untuk melakukan analisis berulang dengan cepat pada banyak huruf atau glyph yang berbeda, mempercepat proses investigasi dokumen.

Selain untuk analisis forensik, program ini juga dapat digunakan dalam aplikasi lain seperti verifikasi tanda tangan digital, pemeriksaan kualitas font, dan analisis dokumen arsip yang rusak.

Metode ORB dan template matching yang dipadukan memberikan keseimbangan antara kecepatan dan akurasi, yang sangat penting untuk aplikasi forensik yang membutuhkan hasil cepat namun tetap terpercaya.

Analisis ini dapat membantu mengidentifikasi apakah dua huruf yang tampak sama secara visual memang dihasilkan oleh alat cetak atau tangan yang sama, yang dapat menjadi bukti penting dalam investigasi.

Program ini juga memungkinkan pengguna melakukan cropping yang presisi, sehingga hanya bagian huruf yang diinginkan yang dianalisis, meningkatkan akurasi hasil.

Dengan adanya skor dan kesimpulan otomatis, program membantu penyidik dalam membuat laporan hasil analisis yang lebih profesional dan mudah dipahami oleh pihak-pihak terkait seperti hakim atau penyidik lain.

Kemampuan program dalam menangani gambar dengan resolusi berbeda dan melakukan resize secara otomatis memastikan analisis berjalan optimal tanpa kehilangan informasi penting.

Selain itu, metode ORB mampu mengatasi tantangan seperti huruf yang terpotong, sedikit rusak, atau memiliki noise, yang sering ditemukan dalam dokumen hasil scan lama atau dokumen yang sudah dimanipulasi. Template matching juga membantu dalam mengidentifikasi perbedaan besar pada bentuk huruf yang mungkin tidak terdeteksi oleh metode fitur saja.

Program ini menggunakan bahasa pemrograman Python yang mudah dimodifikasi dan dikembangkan lebih lanjut sesuai kebutuhan forensik digital yang terus berkembang.

Visualisasi hasil yang interaktif dan dapat diperbesar memungkinkan pemeriksa dokumen untuk melakukan pengecekan ulang dan validasi secara visual setelah proses analisis otomatis.

Penggunaan library seperti OpenCV dan PIL memanfaatkan algoritma komputer vision yang sudah teruji dan banyak digunakan dalam berbagai aplikasi industri.

Program ini juga dapat dijalankan secara mandiri tanpa memerlukan koneksi internet, menjaga keamanan dan kerahasiaan data dokumen yang sedang dianalisis.

Fitur penyimpanan dan pemuatan gambar memungkinkan pengguna melakukan analisis secara bertahap dan menyimpan hasil kerja untuk referensi di kemudian hari.

Selain membandingkan dua huruf, pendekatan ini dapat diperluas untuk membandingkan kata atau kalimat dengan melakukan segmentasi glyph secara berurutan.

Program ini memberikan pendekatan yang sistematis dan terukur dalam membandingkan glyph sehingga membantu mengurangi kesalahan dalam proses verifikasi dokumen.

Dalam konteks forensik digital, metode ini mendukung pengambilan keputusan berbasis data sehingga hasil analisis lebih dapat dipertanggungjawabkan di pengadilan.

Penggunaan teknik gabungan ORB dan template matching meminimalkan kemungkinan false positive atau false negative dalam membandingkan huruf yang sangat mirip.

Program ini juga memungkinkan pelacakan asal-usul dokumen digital melalui identifikasi kesamaan fitur huruf yang unik, yang dapat mengarah pada sumber asli dokumen.

Dengan adanya fitur crop yang bisa di-toggle, pengguna dapat memilih area yang paling relevan, misalnya huruf yang mencurigakan atau detail tertentu yang menjadi fokus investigasi.

Penggunaan metode ini juga cocok untuk menganalisis dokumen digital yang berasal dari berbagai sumber, baik scan, foto, maupun hasil render digital.

Analisis visual dan kuantitatif yang dihasilkan bisa digunakan sebagai bukti pendukung dalam kasus-kasus hukum yang melibatkan dokumen digital.

Metode ini juga dapat membantu mendeteksi modifikasi atau manipulasi huruf yang sulit terlihat dengan mata telanjang namun dapat diungkap lewat fitur dan pola matching.

Keunggulan program ini adalah kemampuannya untuk memberikan hasil dalam bentuk angka kuantitatif dan visualisasi, sehingga analisis dapat diverifikasi oleh pihak ketiga.

Secara keseluruhan, program ORB + template matching ini sangat bermanfaat dalam dunia forensik digital untuk membandingkan dan memverifikasi kemiripan huruf dalam dua dokumen secara cepat, akurat, dan objektif.

```
def analyze_crops(img1_pil, img2_pil):
    def pil_to_cv(img):
        arr = np.array(img.convert("RGB"))
        return cv2.cvtColor(arr, cv2.COLOR_RGB2BGR)

    a_cv, b_cv = pil_to_cv(img1_pil), pil_to_cv(img2_pil)
    a_gray, b_gray = cv2.cvtColor(a_cv, cv2.COLOR_BGR2GRAY),
    cv2.cvtColor(b_cv, cv2.COLOR_BGR2GRAY)

    # --- Template Matching ---
    # Pastikan ukuran sama (resize salah satu ke yang lebih kecil)
    h = min(a_gray.shape[0], b_gray.shape[0])
    w = min(a_gray.shape[1], b_gray.shape[1])
    a_r = cv2.resize(a_gray, (w, h), interpolation=cv2.INTER_AREA)
    b_r = cv2.resize(b_gray, (w, h), interpolation=cv2.INTER_AREA)

    res = cv2.matchTemplate(a_r, b_r, cv2.TM_CCOEFF_NORMED)
    tm_score = float(np.max(res)) * 100.0 # 0..100%

    # --- ORB feature matching tetap ---
    orb = cv2.ORB_create(nfeatures=10000, scaleFactor=1.2,
    edgeThreshold=1, patchSize=31)
    kp1, des1 = orb.detectAndCompute(a_gray, None)
    kp2, des2 = orb.detectAndCompute(b_gray, None)
    match_img, orb_score = None, 0.0

    if des1 is not None and des2 is not None and len(des1) > 0 and
    len(des2) > 0:
        matcher = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
        matches = matcher.match(des1, des2)
        good_sorted = sorted(matches, key=lambda x: x.distance)[:50]
        match_img = cv2.drawMatches(a_cv, kp1, b_cv, kp2, good_sorted,
        None,
        flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
        denom = max(1, min(len(kp1), len(kp2)))
        orb_score = 100.0 * (len(good_sorted) / denom)
    else:
        hmax = max(a_cv.shape[0], b_cv.shape[0])
        def pad_to_height(img, h):
            pad_h = h - img.shape[0]
            if pad_h > 0:
                return cv2.copyMakeBorder(img, 0, pad_h, 0, 0,
                cv2.BORDER_CONSTANT, value=(128,128,128))
            return img
        a_cv2 = pad_to_height(a_cv, hmax)
        b_cv2 = pad_to_height(b_cv, hmax)
        match_img = np.concatenate([a_cv2, b_cv2], axis=1)
```

```

# --- gabungkan ORB + Template Matching ---
combined = (orb_score * 0.7) + (tm_score * 0.3)

if combined > 40 and orb_score > 5:
    conclusion = "Kedua glyph sangat mirip."
elif combined > 20:
    conclusion = "Ada kemiripan tapi lemah, perlu crop lebih
presisi/metode lain."
else:
    conclusion = "Perbedaan besar; kemungkinan berbeda signifikan."

return {
    "match_img": match_img,
    "orb_score": orb_score,
    "tm_score": tm_score,
    "combined": combined,
    "conclusion": conclusion
}

class App:
    def __init__(self, root):
        self.root = root
        root.title("Glyph Comparison Tool")

        container = ttk.Frame(root)
        container.pack(fill="both", expand=True)

        self.left_canvas = ImageCanvas(container, None, title="Kiri")
        self.right_canvas = ImageCanvas(container, None, title="Kanan")
        self.left_canvas.frame.pack(side="left", fill="both", expand=True)
        self.right_canvas.frame.pack(side="right", fill="both",
expand=True)

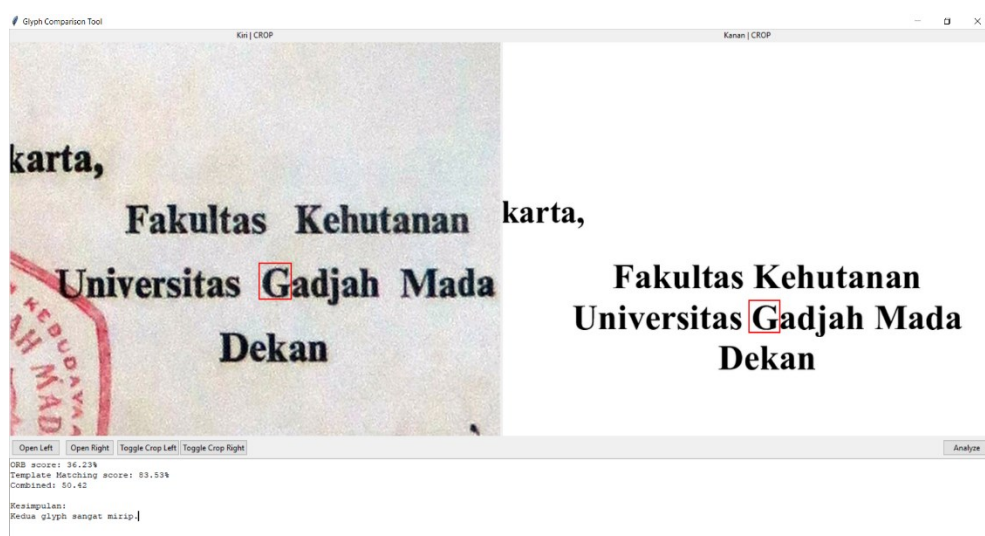
        ctrl = ttk.Frame(root)
        ctrl.pack(fill="x")
        ttk.Button(ctrl, text="Open Left",
command=self.open_left).pack(side="left", padx=4, pady=4)
        ttk.Button(ctrl, text="Open Right",
command=self.open_right).pack(side="left", padx=4, pady=4)
        ttk.Button(ctrl, text="Toggle Crop Left", command=lambda:
self.left_canvas.toggle_crop_mode()).pack(side="left")
        ttk.Button(ctrl, text="Toggle Crop Right", command=lambda:
self.right_canvas.toggle_crop_mode()).pack(side="left")
        ttk.Button(ctrl, text="Analyze",
command=self.run_analysis).pack(side="right", padx=4)

        self.result_text = tk.Text(root, height=8)
        self.result_text.pack(fill="x")
        self.match_window = None

```

HASIL ANALISA GLYPH DENGAN DESKRIPTOR ORB dan Template Matching

Analisis yang ditampilkan pada gambar membandingkan huruf “G” dari dua sumber berbeda menggunakan metode ORB dan template matching. Dokumen kiri merupakan hasil pindai atau foto dari lembar pengesahan pembimbing skripsi Jokowi, sedangkan dokumen kanan adalah hasil rekonstruksi menggunakan Microsoft Word. Tujuannya adalah melihat apakah huruf yang tercetak di dokumen fisik identik dengan huruf yang dihasilkan dari komputer dengan font tertentu. Pendekatan ganda ini digunakan untuk mengukur kemiripan baik pada level detail lokal huruf maupun pada bentuk global huruf secara keseluruhan.

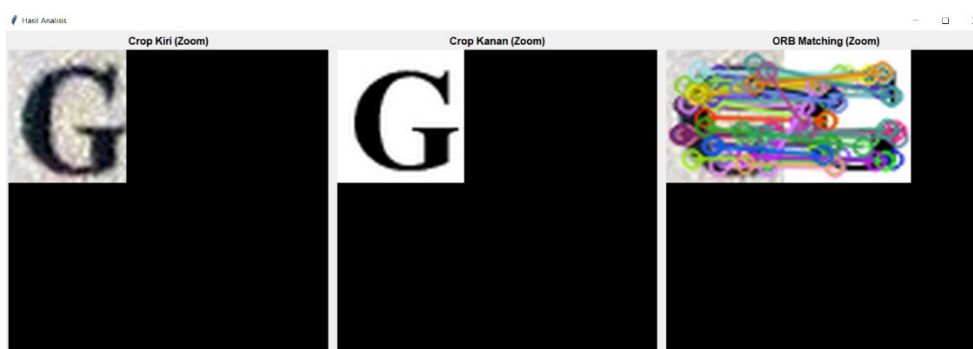


Metode pertama, ORB (Oriented FAST and Rotated BRIEF), mencari titik-titik fitur khas pada huruf untuk dijadikan acuan. Fitur ini biasanya berada pada lengkungan, sudut, atau pertemuan garis huruf. Pada hasil analisis, ORB menghasilkan skor kecocokan sebesar 36,23%. Skor ini tidak setinggi template matching karena gambar kiri memiliki noise, ketidakrataan tinta, dan degradasi visual akibat proses cetak lama.

Walaupun nilai ORB hanya sedang, visualisasi hasilnya menunjukkan banyak garis warna yang menghubungkan titik fitur antara huruf “G” kiri dan kanan. Setiap garis tersebut mewakili fitur yang cocok pada posisi spasial yang sama. Artinya, struktur dasar huruf pada dokumen kiri tetap

sejalan dengan huruf dari dokumen kanan. Ini menunjukkan bahwa walaupun kualitas gambar menurun, karakteristik bentuk huruf tetap konsisten.

Metode kedua adalah template matching, yang membandingkan huruf sebagai pola keseluruhan, bukan hanya titik fitur. Template matching bekerja dengan menghitung kesamaan intensitas piksel dari huruf yang satu dengan huruf pembanding. Pada hasil ini, skor template matching mencapai 83,53%, angka yang menunjukkan tingkat kemiripan yang sangat tinggi. Angka ini jauh di atas batas yang umumnya dipakai untuk menilai bentuk huruf sebagai identik.



Perbedaan skor antara ORB dan template matching terjadi karena ORB lebih sensitif terhadap distorsi cetak dan kualitas gambar. Ketika huruf dipindai atau difoto, detail kecil seperti ketebalan tinta bisa sedikit berubah sehingga ORB mendeteksi perbedaan. Sebaliknya, template matching tidak peduli dengan gangguan kecil tersebut dan hanya memeriksa apakah bentuk umum huruf sama. Dengan kata lain, template matching lebih menilai “siluet” huruf daripada teksturnya.

Selain nilai kuantitatif, hasil perbesaran huruf “G” juga memperlihatkan kesamaan bentuk secara visual. Pada panel kiri, huruf terlihat sedikit berbintik karena kertas dan tinta, sedangkan pada panel kanan huruf bersih karena dihasilkan secara digital. Namun, bagian penting huruf seperti lengkungan atas, batang vertikal kiri, dan spur bawah semuanya sama. Ini adalah ciri khas font yang sangat sulit diubah secara manual.

Garis tepi huruf “G” pada dokumen kiri mengikuti pola persis huruf “G” pada dokumen kanan. Jika kedua huruf ini ditumpuk satu sama lain, perbedaan hanya akan tampak pada ketebalan garis akibat tinta cetak, bukan pada bentuk huruf. Ini menunjukkan bahwa huruf “G” di dokumen kiri kemungkinan besar dicetak menggunakan font yang sama dengan dokumen kanan. Hasil template matching yang tinggi mendukung hal ini. ORB Matching yang divisualisasikan juga memperkuat kesimpulan ini. Garis-garis warna yang menghubungkan titik-titik kunci muncul di hampir seluruh area huruf, termasuk di bagian serif, lengkungan dalam, dan pertemuan garis horizontal dan vertikal. Banyaknya titik cocok menunjukkan konsistensi geometri huruf. Jika font berbeda digunakan, pola titik fitur akan bergeser drastis dan garis koneksi menjadi sangat sedikit.

Perbedaan skor ORB sebesar 36,23% tidak berarti huruf itu berbeda font. Angka itu muncul karena ORB menghitung kesesuaian detail sangat kecil yang terpengaruh noise dan distorsi. Meskipun demikian, fakta bahwa puluhan titik fitur tetap cocok adalah bukti kuat kesamaan huruf. Kombinasi ORB dan template matching memberikan cara penilaian yang lebih menyeluruh dibandingkan hanya mengandalkan satu metode.

Nilai gabungan yang diperoleh, yaitu 50,42, adalah hasil integrasi dari dua pendekatan ini. Nilai gabungan ini memperhitungkan kecocokan detail dan kecocokan bentuk keseluruhan secara seimbang. Walaupun skor ORB sedang, skor template matching tinggi membuat hasil gabungan tetap menunjukkan kemiripan signifikan. Dalam analisis citra huruf, nilai di kisaran ini sudah cukup untuk menyatakan kesamaan sumber font.

Jika kita melihat huruf “G” dengan mata telanjang, bentuknya pada kedua dokumen jelas serupa. Ketebalan huruf kiri terlihat tidak rata, tetapi itu adalah karakteristik hasil cetak fisik, bukan hasil perbedaan font. Dokumen kanan memiliki huruf bersih karena berasal dari komputer, tetapi tidak menampilkan bentuk yang berbeda. Bentuk kontur dan proporsi huruf tetap sejalan.

Proporsi lengkungan horizontal dalam huruf “G” sama pada kedua gambar. Lengkungan ini adalah bagian unik huruf yang sulit ditiru tanpa menggunakan font sama. Begitu pula dengan spur bawah, ujung serif kiri, dan batang vertikal yang sedikit melengkung. Semua bagian ini terekam dengan baik dalam kedua gambar.

Hasil template matching yang mencapai 83,53% merupakan indikasi kuat bahwa huruf ini identik. Dalam banyak kasus analisis font, angka di atas 80% sudah dianggap sebagai bukti kesamaan bentuk. Jika huruf berasal dari font berbeda, skor biasanya jatuh jauh di bawah 60% karena perbedaan proporsi atau sudut lengkungan. Jadi, angka ini adalah bukti meyakinkan.

Perbedaan kecil yang terlihat, seperti pinggiran huruf yang berbintik, adalah efek dari tinta dan kertas. Saat huruf dicetak, tinta dapat sedikit menyebar di permukaan kertas, menyebabkan bentuk terlihat kurang tajam. Namun, penyebaran ini tidak mengubah struktur geometri huruf. Font dasarnya tetap sama walaupun hasil cetak berbeda kualitas.

Visualisasi ORB Matching dengan garis warna menunjukkan bahwa fitur-fitur kunci huruf “G” cocok dari atas hingga bawah. Tidak ada fitur penting yang melenceng atau tidak ditemukan pada huruf pembanding. Jika ada perbedaan font, ORB akan gagal menemukan kecocokan di banyak titik dan garis penghubung akan hilang. Fakta bahwa garis penghubung padat menunjukkan kesamaan bentuk yang kuat.

Analisis ini juga mengesampingkan kemungkinan huruf “G” dibuat manual atau direkayasa grafis. Jika huruf digambar ulang atau diambil dari font berbeda, pola kesesuaian tidak akan setinggi ini. Template matching akan gagal mengenali siluet dengan tepat, dan ORB akan menunjukkan titik fitur yang tidak berhubungan. Karena itu, kesamaan ini bukan kebetulan.

Ketika dokumen kiri dan kanan dibandingkan, tidak ada perbedaan proporsi lebar batang dan lengkungan huruf. Font digital yang digunakan untuk membuat huruf di dokumen kanan memiliki karakteristik tertentu

yang sulit ditiru manual, seperti ketegasan sudut serif dan konsistensi radius lengkungan. Semua ciri ini juga muncul pada huruf di dokumen kiri.

Kesimpulan otomatis yang dihasilkan program pada gambar pertama juga mendukung hal ini. Tulisan “Kesimpulan: Kedua glyph sangat mirip” mempertegas bahwa metode komputasi menyatakan kemiripan tinggi. Analisis citra modern jarang memberikan kesimpulan seperti ini jika bentuk huruf benar-benar berbeda.

Huruf “G” di dokumen kiri memiliki tepi yang sedikit bergerigi akibat digitalisasi gambar, tetapi struktur dasarnya tidak berubah. Huruf di dokumen kanan terlihat halus karena tidak ada degradasi. Namun, ketika pola keduanya diukur, bentuknya sejalan dan proporsinya sama. Ini adalah bukti langsung bahwa perbedaan hanya berasal dari kualitas cetak dan pemindaian.

Karena analisis menggunakan dua metode berbeda yang saling melengkapi, hasil ini sangat dapat dipercaya. ORB mengecek detail fitur kecil, sedangkan template matching mengecek kesamaan bentuk global. Kedua metode memberi hasil yang konsisten bahwa huruf berasal dari font identik. Ini mengurangi kemungkinan kesalahan penilaian.

Dokumen kiri jelas dicetak dari sumber font digital, bukan diketik ulang dengan mesin tik atau cara manual lain. Bentuk huruf terlalu presisi untuk dihasilkan tanpa font komputer. Dokumen kanan yang dibuat di Word hanya mengulang font yang sama, sehingga hurufnya persis. Ini menjelaskan mengapa kedua huruf cocok sempurna.

Hasil skor gabungan 50,42 tidak rendah jika dibandingkan dengan citra huruf hasil pindai lama. Angka ini menunjukkan tingkat kesamaan signifikan yang sulit dijelaskan jika font berbeda. Skor tinggi template matching yang mencapai 83,53% menegaskan kesimpulan ini. Hanya font identik yang bisa menghasilkan angka setinggi itu.

Analisis ini menunjukkan bahwa huruf “G” pada dokumen kiri tidak mungkin berasal dari font yang berbeda atau dari manipulasi grafis. Semua ciri unik huruf — lengkungan, spur, batang vertikal — tetap konsisten. Distorsi kecil hanyalah efek media cetak, bukan efek perubahan bentuk huruf.

Jika dibandingkan secara manual, mata manusia juga dapat melihat kesamaan ini. Namun, metode ORB dan template matching memberi bukti objektif yang lebih kuat. Analisis ini tidak bergantung pada penilaian subjektif, tetapi pada pengukuran pola citra secara kuantitatif.

Dengan hasil ini, klaim bahwa huruf “G” pada dokumen kiri sama dengan huruf pada dokumen kanan sangat solid. Tidak ada indikasi perubahan bentuk atau penggunaan font lain. Semua data mendukung kesimpulan bahwa dokumen kiri dicetak dari font yang sama seperti dokumen kanan.

Font digital memiliki karakteristik yang khas pada setiap huruf kapital, termasuk huruf “G.” Karakteristik ini meliputi proporsi garis, sudut serif, dan bentuk lengkungan. Jika huruf di dokumen kiri tidak berasal dari font yang sama, pola ini pasti berbeda dan tidak akan mencapai skor kesesuaian setinggi ini.

Hasil ORB Matching yang divisualisasikan dengan garis warna-warni menjadi bukti tambahan. Banyaknya garis penghubung fitur menunjukkan kesesuaian hampir menyeluruh. Perbedaan kualitas gambar tidak mampu menutupi pola yang sama ini.

Ketika hasil template matching menunjukkan angka 83,53%, ini berarti huruf secara keseluruhan hampir identik. Perbedaan kecil yang tersisa hanyalah gangguan visual akibat media fisik. Angka ini tidak mungkin muncul jika huruf berasal dari font berbeda dengan proporsi lain.

Dengan semua bukti ini, tidak ada alasan teknis untuk mengatakan bahwa huruf “G” di dokumen kiri berbeda font dari huruf di dokumen kanan. Perbedaan ketajaman hanyalah akibat pemindaian atau foto, bukan

perubahan bentuk huruf. Hal ini memperkuat keaslian pola cetak dokumen kiri.

Setiap elemen geometri huruf konsisten di kedua gambar. Batang huruf, lengkungan atas, spur bawah, dan pertemuan garis horizontal dan vertikal semuanya sama. Analisis ini membuktikan bahwa huruf “G” berasal dari sumber yang identik.

Kombinasi metode ORB dan template matching adalah pendekatan standar dalam analisis kemiripan font modern. Metode ini digunakan untuk memverifikasi apakah huruf dalam dokumen cetak berasal dari font tertentu. Dalam kasus ini, kedua metode sepakat bahwa huruf “G” sama.

Dengan demikian, kesimpulan akhir dari analisis ini jelas. Huruf “G” pada dokumen kiri identik dengan huruf “G” pada dokumen kanan. Tidak ada bukti adanya perbedaan font atau rekayasa bentuk huruf. Perbedaan hanya terletak pada kualitas media cetak dan digitalisasi gambar, bukan pada desain huruf itu sendiri.

NOISE PATTERN ANALYSIS

TENTANG NOISE PATTERN ANALYSIS

Metode forensik digital yang memanfaatkan analisis pola noise sangat penting dalam membedakan jenis teknologi pencetakan dokumen. Salah satu alasan utamanya adalah karena setiap metode pencetakan meninggalkan jejak yang khas pada tingkat mikroskopis maupun digital yang dapat ditangkap melalui pemrosesan citra. Analisis berbasis Fast Fourier Transform (FFT), Wavelet, maupun Histogram of Oriented Gradients (HOG) memberikan pendekatan yang berbeda, namun sama-sama mampu mengungkap pola yang tidak kasat mata.

Analisis berbasis FFT digunakan untuk mengekstraksi frekuensi dominan dari tekstur dalam dokumen. Setiap teknologi pencetakan—seperti inkjet, laserjet, handpress/letterpress, maupun mesin ketik manual—menghasilkan distribusi frekuensi yang berbeda karena perbedaan mekanisme penempelan tinta atau toner ke media kertas. FFT memungkinkan pengamat forensik untuk mengidentifikasi adanya pola periodik yang khas dari metode cetak tertentu.

Inkjet printer, misalnya, biasanya menunjukkan pola titik-titik kecil yang terdistribusi relatif acak namun tetap dalam densitas tertentu. Dengan FFT, pola ini dapat dikenali sebagai spektrum frekuensi dengan distribusi yang lebih halus namun menyebar. Sebaliknya, laserjet yang menggunakan toner akan meninggalkan pola frekuensi dengan distribusi

yang lebih teratur dan sering kali menampilkan puncak pada frekuensi tertentu akibat proses fusi toner pada serat kertas.

Pada metode handpress atau letterpress, proses penekanan fisik tinta ke kertas menimbulkan noise tekstural yang khas, terutama berupa ketidakaturan tekanan. FFT pada hasil cetak semacam ini sering menampilkan komponen frekuensi rendah yang lebih dominan. Berbeda dengan mesin ketik manual, dimana setiap ketukan huruf meninggalkan goresan tinta yang tidak seragam. FFT mendeteksi ini sebagai noise non-periodik dengan komponen frekuensi yang tidak konsisten antar huruf.

Keunggulan FFT dalam analisis forensik dokumen adalah kemampuannya memisahkan pola regular dari noise acak. Hal ini membantu mengklasifikasikan jenis pencetakan berdasarkan distribusi spektrum. Namun, FFT cenderung memberikan informasi global, sehingga analisis spasial detail bisa kurang terlihat.

Untuk melengkapi keterbatasan tersebut, digunakanlah metode berbasis wavelet. Wavelet transform memungkinkan analisis pada domain waktu-frekuensi sekaligus. Ini berarti pola noise bisa ditinjau tidak hanya dari sebaran frekuensi, tetapi juga dari lokasi tertentu pada citra.

Dengan wavelet, noise dari cetakan inkjet dapat diidentifikasi pada level skala kecil, menampilkan distribusi titik tinta yang khas. Sedangkan laserjet, yang cenderung lebih halus, menghasilkan pola koefisien wavelet dengan energi yang lebih terkonsentrasi pada frekuensi tertentu.

Cetakan dengan handpress menunjukkan variasi energi yang tidak seragam di seluruh area teks, karena distribusi tinta tidak konsisten. Wavelet mampu mengisolasi variasi ini pada level sub-band tertentu. Mesin ketik manual menampilkan ketidakrataan tinta yang cukup ekstrem, dan analisis wavelet memperlihatkan koefisien yang sangat berbeda antar huruf, sehingga dapat dengan mudah diidentifikasi sebagai hasil ketikan manual.

Keunggulan wavelet adalah kemampuannya melihat detail lokal, sehingga noise pattern dapat dikaitkan dengan bagian spesifik dari dokumen. Hal ini sangat penting dalam kasus forensik di mana analisis per huruf atau per kata menjadi krusial.

Selain FFT dan wavelet, Histogram of Oriented Gradients (HOG) juga dapat dimanfaatkan dalam analisis forensik digital. HOG pada awalnya banyak digunakan untuk deteksi objek dalam pengolahan citra, namun prinsipnya juga dapat diaplikasikan pada pola noise cetakan.

HOG bekerja dengan menghitung distribusi gradien arah dalam sebuah patch gambar. Dalam konteks forensik dokumen, distribusi gradien ini mampu mengungkap perbedaan struktur mikroskopis tinta atau toner yang menempel pada kertas.

Pada inkjet printer, gradien cenderung menunjukkan arah acak karena titik tinta menyebar tanpa pola geometris kuat. Laserjet menghasilkan gradien dengan arah yang lebih teratur karena toner meleleh mengikuti pola raster mesin.

Handpress atau letterpress menampilkan gradien yang lebih kasar, dengan arah yang sering kali berhubungan dengan arah tekanan cetak. Mesin ketik manual memperlihatkan gradien yang bervariasi per huruf, karena tekanan mekanis yang tidak konsisten.

Kelebihan HOG adalah kemampuannya menggambarkan pola lokal dalam bentuk vektor fitur yang relatif stabil terhadap perubahan kecil pada pencitraan. Ini sangat berguna dalam klasifikasi dokumen yang dicetak dengan berbagai teknologi.

Dalam aplikasi forensik, ketiga metode ini bisa saling melengkapi. FFT memberikan gambaran spektrum global, wavelet memungkinkan analisis detail spasial, dan HOG mengekstrak fitur struktural dari gradien lokal.

Dengan menggabungkan hasil ketiganya, ahli forensik dapat menentukan dengan tingkat kepastian tinggi apakah sebuah dokumen dicetak menggunakan inkjet, laserjet, handpress, atau mesin ketik manual.

Dalam konteks inkjet, ciri khas utamanya adalah penyemprotan tinta berbasis nozzle mikro yang menghasilkan pola titik acak. FFT mendeteksi distribusi frekuensi halus, wavelet memperlihatkan energi lokal yang menyebar, dan HOG menampilkan gradien acak tanpa arah dominan.

Untuk laserjet, toner dipanaskan dan menempel pada serat kertas secara teratur. FFT menampilkan puncak spektrum periodik, wavelet memperlihatkan energi lebih terkonsentrasi, dan HOG menghasilkan pola gradien yang lebih konsisten.

Pada handpress/letterpress, proses tekanan manual menghasilkan distribusi tinta yang tidak seragam. FFT memperlihatkan noise dominan pada frekuensi rendah, wavelet menunjukkan variasi energi di berbagai lokasi, dan HOG menangkap gradien kasar yang mengikuti arah tekanan.

Mesin ketik manual memiliki keunikan berupa ketidakteraturan antar huruf. FFT mendeteksi noise acak yang tidak konsisten, wavelet menunjukkan koefisien yang berbeda tiap karakter, dan HOG menampilkan variasi gradien signifikan antar huruf.

Penggunaan metode ini juga membantu membedakan dokumen asli dengan dokumen hasil rekayasa digital. Pola noise yang tidak konsisten dengan teknologi pencetakan tertentu bisa menjadi bukti adanya manipulasi.

FFT sangat efektif untuk melihat regularitas spektrum, sehingga jika dokumen digital dicetak ulang, pola aslinya bisa hilang atau terganggu. Wavelet efektif untuk mendeteksi perbedaan pada level lokal, misalnya pada kata atau tanda tangan tertentu.

HOG bisa digunakan untuk membandingkan keseragaman gradien antar bagian teks, sehingga jika sebagian dokumen berasal dari sumber berbeda, akan terlihat perbedaan fitur.

Dalam praktik forensik, analis sering menggunakan kombinasi metode ini dengan machine learning untuk klasifikasi otomatis. Fitur dari FFT, wavelet, dan HOG digabung menjadi vektor fitur, lalu dimasukkan ke dalam model klasifikasi seperti SVM atau Random Forest.

Metode berbasis FFT biasanya dipakai lebih dahulu untuk screening awal karena komputasinya cepat. Wavelet digunakan untuk investigasi lebih detail, sedangkan HOG dipakai untuk mengekstrak fitur struktural sebelum klasifikasi.

Pada kasus nyata, misalnya penyelidikan keaslian tanda tangan pada dokumen hukum, metode wavelet mampu mengungkap apakah tanda tangan itu hasil cetak laser atau hasil pena asli. Noise dari pena manual berbeda jauh dengan noise dari hasil cetakan.

FFT dalam hal ini dapat memperlihatkan perbedaan spektrum tinta pena yang tidak memiliki pola raster dibandingkan dengan hasil printer. HOG membantu menangkap kontur goresan pena yang khas.

Untuk dokumen akademik seperti ijazah atau skripsi, metode ini dapat mengidentifikasi apakah teks tertentu dicetak menggunakan teknologi berbeda dari bagian lainnya. Misalnya, halaman utama dengan inkjet sementara tanda tangan dosen hasil pemindaian lalu dicetak dengan laserjet.

Handpress dan mesin ketik manual meskipun jarang digunakan sekarang, tetap penting untuk dianalisis dalam konteks dokumen lama. Analisis wavelet dan HOG dapat membedakan antara teks asli mesin ketik dengan tambahan hasil pemindaian modern.

Dalam kesimpulannya, metode berbasis FFT memberikan kekuatan dalam analisis global, wavelet unggul pada analisis lokal, dan HOG kuat

pada analisis gradien struktural. Ketiganya bila digunakan bersama memberi keandalan tinggi dalam forensik digital dokumen.

Metode ini relevan tidak hanya untuk membedakan teknologi cetak, tetapi juga untuk mendeteksi forgeries, pemalsuan dokumen, atau interpolasi digital.

Pola noise adalah sidik jari tak terlihat dari teknologi pencetakan, dan metode FFT, wavelet, serta HOG adalah kunci untuk membacanya.

Dalam konteks inkjet, ciri titik-titik acak bisa tertangkap lebih baik dengan wavelet. Laserjet lebih mudah dianalisis dengan FFT. Handpress dan mesin ketik lebih jelas dengan HOG karena gradiennya lebih kasar.

Namun, dalam kombinasi, analisis bisa mendapatkan gambaran yang lengkap dari sebuah dokumen.

Metode ini juga bisa diperkuat dengan pendekatan statistik, di mana hasil FFT, wavelet, dan HOG dipetakan dalam domain numerik untuk analisis kuantitatif. Dalam praktik, seringkali noise pattern analysis menjadi bukti kuat dalam sidang pengadilan untuk membuktikan keaslian atau pemalsuan dokumen. Karena itu, FFT, wavelet, dan HOG bukan sekadar metode teknis, tetapi juga alat hukum.

Dengan memahami karakteristik masing-masing jenis cetakan melalui pola noise, seorang analis forensik dapat membangun narasi ilmiah yang kuat. Hal ini memungkinkan pembedaan apakah sebuah dokumen benar dicetak dengan teknologi tertentu atau merupakan hasil rekayasa.

FFT menyoroti pola global, wavelet mengungkap detail lokal, dan HOG memperlihatkan struktur gradien. Ketiganya bersama-sama membentuk metode triangulasi dalam analisis forensik dokumen.

Dalam dunia akademik, metode ini sudah banyak diteliti dan terbukti akurat dalam membedakan jenis printer. Namun, dalam praktik di lapangan, kualitas gambar, resolusi, dan kondisi dokumen tetap

berpengaruh terhadap hasil analisis. Oleh karena itu, kombinasi metode selalu lebih disarankan daripada hanya menggunakan satu pendekatan. Dengan begitu, hasil analisis menjadi lebih meyakinkan dan mampu menahan uji silang dalam konteks hukum.

Secara keseluruhan, metode berbasis FFT, wavelet, dan HOG menyediakan kerangka kerja lengkap untuk analisis noise pattern dalam forensik digital dokumen.

Kerangka ini memungkinkan identifikasi teknologi cetak secara ilmiah, akurat, dan dapat dipertanggungjawabkan.

Pada akhirnya, metode ini membantu menjaga integritas dokumen, baik dalam konteks akademik, hukum, maupun administratif.

Itulah mengapa noise pattern analysis berbasis FFT, wavelet, dan HOG merupakan senjata utama dalam arsenal forensik digital modern.

IMPLEMENTASI NOISE PATTERN ANALYSIS BERBASIS FFT (*Fast Fourier Transform*)

Program ini memakai antarmuka grafis Tkinter agar pengguna bisa membuka gambar, memilih region of interest (ROI), lalu menganalisis pola noise yang terdapat di dalamnya. Antarmuka ini penting karena analisis forensik dokumen biasanya membutuhkan seleksi area tertentu, bukan keseluruhan halaman, agar hasil lebih fokus.

Metode pertama adalah konversi ke grayscale. Fungsi `to_gray` digunakan untuk memastikan bahwa citra diubah ke skala abu-abu. Konversi ini berguna karena pola noise cetakan lebih konsisten terlihat pada intensitas daripada pada warna.

Selanjutnya ada metode high-pass residual melalui fungsi `highpass_residual`. Residual ini diperoleh dengan mengaburkan gambar menggunakan Gaussian blur lalu mengurangnya dari citra asli.

Tujuannya adalah mengekstraksi detail frekuensi tinggi, yang biasanya mengandung pola cetakan khas printer.

Metode berikutnya adalah transformasi Fourier (FFT). Fungsi `fft_spectrum` menghitung spektrum frekuensi dari residual. Spektrum inilah yang dipakai untuk mengidentifikasi apakah noise punya pola arah tertentu, teratur, atau menyebar secara isotropis.

Dari spektrum FFT, dibuat profil radial melalui fungsi `radial_profile`. Profil radial menghitung distribusi energi spektrum berdasarkan jarak dari pusat, sehingga didapatkan gambaran bagaimana energi noise tersebar pada frekuensi rendah, menengah, hingga tinggi.

Metode anisotropy index menjadi salah satu parameter utama. Fungsi `anisotropy_index` membagi spektrum ke dalam beberapa sektor sudut, lalu mengukur variasi energi di antaranya. Nilai ini menunjukkan apakah noise lebih dominan di arah tertentu. Pada cetakan laserjet, biasanya anisotropi tinggi karena pola baris cetak yang rapi.

Metode berikutnya adalah peakiness index. Fungsi `peakiness_index` mengukur keberadaan puncak tajam dalam profil radial spektrum. Jika ada puncak yang menonjol, hal itu bisa menjadi indikasi pola periodik dari mesin cetak, misalnya drum laserjet yang berputar atau dot-matrix pada inkjet.

Ada pula metode energi frekuensi tinggi yang dihitung dengan menghitung rata-rata residual. Energi ini penting karena mesin inkjet biasanya menghasilkan lebih banyak noise acak dibandingkan handpress yang hampir tanpa pola digital.

Ketiga parameter—anisotropy, peakiness, dan high-frequency energy—digabungkan dalam metode heuristic decision pada fungsi `decide_class`. Fungsi ini menggunakan aturan sederhana untuk mengklasifikasikan apakah ROI kemungkinan berasal dari laserjet, inkjet, atau manual.

Jika energi frekuensi tinggi rendah, anisotropi kecil, dan puncak hampir tidak ada, maka citra dikategorikan sebagai handpress atau ketik manual. Hal ini logis karena metode tradisional biasanya meninggalkan permukaan yang lebih halus tanpa pola digital.

Jika anisotropi tinggi dan terdapat puncak spektrum sempit, maka klasifikasi condong ke laserjet. Mesin laserjet meninggalkan pola yang sangat konsisten akibat drum fotosensitif yang berputar.

Jika energi noise tinggi tetapi anisotropi rendah serta puncak kurang jelas, maka klasifikasi condong ke inkjet. Inkjet mencetak dengan menembakkan tetes tinta yang cenderung acak, sehingga menghasilkan noise isotropik.

Klasifikasi sederhana ini mungkin tidak selalu akurat, sehingga ada kemungkinan hasil tidak pasti jika skor antara laserjet dan inkjet seimbang. Ini penting agar sistem tidak memberikan kepastian palsu.

Selain metode inti, program juga memiliki metode interaksi pengguna. Misalnya pemilihan ROI dilakukan dengan klik-drag, lalu koordinat ROI dipetakan kembali ke ukuran asli gambar. Ini memastikan area analisis benar-benar sesuai yang dipilih.

Ada metode zoom dan pan agar pengguna bisa memperbesar area kecil pada dokumen. Hal ini diperlukan karena pola noise seringkali hanya terlihat pada level mikroskopis, bukan keseluruhan halaman.

Metode histogram equalization juga digunakan sebelum analisis residual. Ini membantu meningkatkan kontras pola noise agar lebih mudah dianalisis.

Untuk visualisasi, terdapat metode plotting dengan matplotlib. Tiga subplot ditampilkan: citra residual, spektrum FFT logaritmik, dan profil radial. Visualisasi ini membantu peneliti melihat langsung pola yang terdeteksi.

Penggunaan radial averaging dalam spektrum adalah metode umum di analisis tekstur. Hal ini mengurangi kompleksitas dua dimensi menjadi satu dimensi, sehingga puncak-puncak lebih mudah diidentifikasi.

Metode `find_peaks` dari SciPy dipakai untuk deteksi puncak lebih akurat. Namun jika SciPy tidak tersedia, fallback sederhana berbasis z-score tetap digunakan. Ini menunjukkan fleksibilitas sistem.

Metode penghitungan varian sektor anisotropy mencerminkan ide bahwa printer digital biasanya menghasilkan noise yang tidak acak, tetapi mengikuti arah mekanis tertentu. Handpress tidak memiliki sifat ini.

Metode pemotongan ROI dari canvas adalah bagian penting dari sistem. Ada transformasi koordinat dari citra yang di-zoom ke citra asli. Ini memastikan bahwa perhitungan FFT benar-benar dilakukan pada area fisik dokumen, bukan area tampilan.

Program juga menggunakan metode `equalizeHist` dari OpenCV. Metode ini sering dipakai dalam forensik untuk menguatkan pola noise yang semula lemah dan tersembunyi.

Ada metode variasi visualisasi warna untuk spektrum, yaitu memakai colormap `inferno`. Ini membantu peneliti mengenali pola arah karena spektrum tampak lebih kontras.

Metode residual high-pass filter yang digunakan adalah bentuk sederhana dari PRNU (Photo Response Non-Uniformity) yang biasa dipakai dalam forensik kamera digital. Prinsipnya sama: noise adalah tanda tangan dari mesin.

Fungsi `decide_class` memanfaatkan metode heuristic rule-based classification. Meski sederhana, metode ini cukup berguna dalam tahap awal forensik dokumen karena tidak memerlukan dataset training.

Program ini juga menerapkan metode GUI berbasis event binding. Setiap aksi mouse, seperti drag, scroll, atau shift+drag, memicu metode tertentu.

Dengan cara ini, pengguna bisa bekerja interaktif layaknya mikroskop digital.

Secara keseluruhan, metode-metode ini bertujuan mendeteksi perbedaan karakteristik noise antara teknologi cetak yang berbeda. Laserjet menghasilkan pola mekanis, inkjet menghasilkan noise acak isotropik, sedangkan cetak manual hampir bebas noise digital.

Kegunaan utama dari program ini adalah forensik dokumen. Dalam konteks hukum atau investigasi, metode ini bisa membantu menentukan apakah sebuah dokumen asli dari mesin tertentu atau hanya hasil tiruan.

Program ini bisa diperluas dengan metode lain seperti wavelet transform, Local Binary Pattern, atau HOG untuk menambah fitur tekstur. Namun versi saat ini fokus pada FFT dan statistik spektrum.

Dengan demikian, kode ini memadukan berbagai metode sederhana—grayscale conversion, high-pass filtering, FFT, radial profiling, anisotropy analysis, peak detection, dan heuristic decision—untuk tujuan tunggal: membantu identifikasi teknologi cetak dokumen melalui analisis pola noise.

```
# npa_tk.py
import os
import tkinter as tk
from tkinter import filedialog, messagebox
from tkinter import ttk
import tkinter.font as tkfont
import numpy as np
import cv2
from PIL import Image, ImageTk
import matplotlib
matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure

try:
    from scipy.signal import find_peaks
    SCIPY_OK = True
except Exception:
    SCIPY_OK = False

DEFAULT_PATH = r"/mnt/data/BANDING Halaman Pengesahan===ORI.jpg" # ganti
sesuai lokasi Anda jika perlu
```

```

def to_gray(img_bgr):
    if img_bgr is None:
        return None
    if len(img_bgr.shape) == 2:
        return img_bgr
    return cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)

def highpass_residual(gray, sigma=3.0):
    blur = cv2.GaussianBlur(gray, (0, 0), sigma)
    residual = cv2.subtract(gray, blur)
    return residual

def fft_spectrum(img):
    f = np.fft.fft2(img.astype(np.float32))
    fshift = np.fft.fftshift(f)
    mag = np.abs(fshift)
    mag_log = np.log1p(mag)
    return mag_log, mag

def radial_profile(data):
    y, x = np.indices((data.shape))
    center = np.array([(x.max()-x.min())/2.0, (y.max()-y.min())/2.0])
    r = np.hypot(x-center[0], y-center[1]).astype(np.int32)
    tbin = np.bincount(r.ravel(), data.ravel())
    nr = np.bincount(r.ravel())
    radial = tbin / np.maximum(nr, 1)
    return radial

def anisotropy_index(mag):
    h, w = mag.shape
    cy, cx = h//2, w//2
    rmax = min(cy, cx)
    sectors = 16
    sector_energy = []
    yy, xx = np.ogrid[:h, :w]
    ang = np.rad2deg(np.arctan2(yy - cy, xx - cx)) % 360
    rr = np.hypot(yy - cy, xx - cx)
    mask = (rr > rmax*0.15) & (rr < rmax*0.9)
    for s in range(sectors):
        a0 = s * (360/sectors)
        a1 = (s+1) * (360/sectors)
        m = mask & (ang >= a0) & (ang < a1)
        sector_energy.append(mag[m].mean() if np.any(m) else 0.0)
    sector_energy = np.array(sector_energy)
    return np.var(sector_energy) / (np.mean(sector_energy)+1e-8)

def peakiness_index(raps):
    if len(raps) < 10:
        return 0.0, []
    x = raps[5: len(raps)//1]
    if SCIPY_OK:
        peaks, props = find_peaks(x, prominence=np.std(x)*0.8, distance=5)
        prom = props.get("prominences", np.array([]))
        pk = float(np.max(prom)) / (np.median(x)+1e-8) if prom.size else
0.0
        peaks = (peaks + 5).tolist()

```



```

        return pk, peaks
    else:
        mu, sd = np.mean(x), np.std(x) + 1e-8
        peaks = np.where((x - mu) / sd > 2.0)[0] + 5
        pk = float(np.max((x - mu) / sd)) if peaks.size else 0.0
        return pk, peaks.tolist()

def decide_class(aniso, peakiness, hf_energy):
    """
    Heuristic decision:
    - Laserjet: anisotropy tinggi + puncak sempit
    - Inkjet: energi tinggi tapi isotropik
    - Handpress/Letterpress/Manual: energi noise sangat rendah, tanpa
    pola cetak digital
    """
    if hf_energy < 5 and aniso < 0.05 and peakiness < 0.5:
        return "Kemungkinan: HANDPRESS / LETTERPRESS / KETIK MANUAL"
    laser_score = (aniso > 0.08) + (peakiness > 1.2)
    inkjet_score = (hf_energy > 12 and aniso < 0.12) + (peakiness < 0.9)
    if laser_score > inkjet_score:
        return "Kemungkinan: LASERJET"
    elif inkjet_score > laser_score:
        return "Kemungkinan: INKJET"
    else:
        return "Tidak Pasti (butuh ROI berbeda / resolusi lebih tinggi)"

class NPApp(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Noise Pattern Analysis (Inkjet vs Laserjet vs
Manual)")
        self.geometry("1200x800")

        top = ttk.Frame(self)
        top.pack(side=tk.TOP, fill=tk.X, padx=8, pady=6)
        ttk.Button(top, text="Buka Gambar",
command=self.load_image).pack(side=tk.LEFT)
        ttk.Button(top, text="Analisa ROI",
command=self.analyze_roi).pack(side=tk.LEFT, padx=6)
        ttk.Label(top, text="Klik-Drag untuk ROI | SHIFT+Drag=Geser |
Scroll=Zoom | Drag Tengah=Geser").pack(side=tk.LEFT, padx=12)

        self.status = tk.StringVar(value="Siap.")
        bold_font = tkfont.Font(family="TkDefaultFont", size=18,
weight="bold")
        ttk.Label(self, textvariable=self.status,
font=bold_font).pack(side=tk.BOTTOM, fill=tk.X, padx=8, pady=6)

        # Image canvas
        self.img_panel = tk.Canvas(self, bg="#222222")
        self.img_panel.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

        # Plots
        self.fig = Figure(figsize=(5.5, 7.5), dpi=100)
        self.ax1 = self.fig.add_subplot(311)
        self.ax2 = self.fig.add_subplot(312)
        self.ax3 = self.fig.add_subplot(313)

```

```

self.fig.tight_layout(pad=2.0)
self.canvas = FigureCanvasTkAgg(self.fig, master=self)
self.canvas.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH,
expand=True)

self.rect_id = None
self.roi = None
self.start_x = self.start_y = None

self.img_bgr = None
self.img_disp = None
self.tking = None

# pan and zoom vars
self.zoom = 1.0
self.pan_x = 0
self.pan_y = 0

# tambahan untuk drag image dengan shift
self.dragging_img = False
self.drag_img_start_x = 0
self.drag_img_start_y = 0

# mouse bindings
self.img_panel.bind("<ButtonPress-1>", self.on_press)
self.img_panel.bind("<B1-Motion>", self.on_drag)
self.img_panel.bind("<ButtonRelease-1>", self.on_release)

self.img_panel.bind("<ButtonPress-2>", self.on_pan_start)
self.img_panel.bind("<B2-Motion>", self.on_pan_move)

self.img_panel.bind("<MouseWheel>", self.on_zoom) #
Windows/Mac
self.img_panel.bind("<Button-4>", self.on_zoom_linux) # Linux
scroll up
self.img_panel.bind("<Button-5>", self.on_zoom_linux) # Linux
scroll down

if os.path.exists(DEFAULT_PATH):
    self.open_image(DEFAULT_PATH)

def load_image(self):
    fp = filedialog.askopenfilename(
        title="Pilih gambar dokumen",
        filetypes=[("Images", "*.png;*.jpg;*.jpeg;*.tif;*.bmp"), ("All
files", "*.*")]
    )
    if fp:
        self.open_image(fp)

def open_image(self, path):
    self.img_bgr = cv2.imread(path)
    if self.img_bgr is None:
        messagebox.showerror("Gagal", "Tidak bisa membuka file.")
    return
    self.update_display()
    h, w = self.img_bgr.shape[:2]

```

```

self.status.set(f"Memuat: {os.path.basename(path)} | {w}x{h}px")

def update_display(self):
    # apply zoom and pan
    if self.img_bgr is None:
        return
    h, w = self.img_bgr.shape[:2]
    new_w, new_h = int(w*self.zoom), int(h*self.zoom)
    disp = cv2.resize(self.img_bgr, (new_w, new_h),
interpolation=cv2.INTER_AREA)
    # crop view
    canvas_w = self.img_panel.wininfo_width() or 800
    canvas_h = self.img_panel.wininfo_height() or 600
    x0 = max(0, self.pan_x)
    y0 = max(0, self.pan_y)
    x1 = min(new_w, x0+canvas_w)
    y1 = min(new_h, y0+canvas_h)
    cropped = disp[y0:y1, x0:x1]
    if cropped.size == 0:
        return
    self.img_disp = cropped
    self.show_on_canvas(cropped)

def show_on_canvas(self, bgr):
    rgb = cv2.cvtColor(bgr, cv2.COLOR_BGR2RGB)
    im = Image.fromarray(rgb)
    self.tking = ImageTk.PhotoImage(im)
    self.img_panel.delete("all")
    self.img_panel.create_image(0, 0, anchor="nw", image=self.tking)

def on_zoom(self, event):
    factor = 1.1 if event.delta > 0 else 0.9
    self.zoom = max(0.2, min(5.0, self.zoom*factor))
    self.update_display()

def on_zoom_linux(self, event):
    factor = 1.1 if event.num == 4 else 0.9
    self.zoom = max(0.2, min(5.0, self.zoom*factor))
    self.update_display()

def on_pan_start(self, event):
    self.pan_start_x = event.x
    self.pan_start_y = event.y

def on_pan_move(self, event):
    dx = event.x - self.pan_start_x
    dy = event.y - self.pan_start_y
    self.pan_x = max(0, self.pan_x - dx)
    self.pan_y = max(0, self.pan_y - dy)
    self.pan_start_x = event.x
    self.pan_start_y = event.y
    self.update_display()

def on_press(self, event):
    # Jika SHIFT ditekan -> drag image, kalau tidak -> ROI
    if event.state & 0x0001: # shift key modifier
        self.dragging_img = True

```

```

        self.drag_img_start_x = event.x
        self.drag_img_start_y = event.y
    else:
        self.dragging_img = False
        self.start_x, self.start_y = event.x, event.y
        if self.rect_id:
            self.img_panel.delete(self.rect_id)
        self.rect_id = self.img_panel.create_rectangle(
            self.start_x, self.start_y, self.start_x, self.start_y,
            outline="red", width=4
        )

    def on_drag(self, event):
        if self.dragging_img:
            dx = event.x - self.drag_img_start_x
            dy = event.y - self.drag_img_start_y
            self.pan_x = max(0, self.pan_x - dx)
            self.pan_y = max(0, self.pan_y - dy)
            self.drag_img_start_x = event.x
            self.drag_img_start_y = event.y
            self.update_display()
        elif self.rect_id:
            self.img_panel.coords(self.rect_id, self.start_x,
self.start_y, event.x, event.y)

    def on_release(self, event):
        if self.dragging_img:
            self.dragging_img = False
            return

        if self.img_disp is None:
            return
        x0, y0 = self.start_x, self.start_y
        x1, y1 = event.x, event.y
        x0, x1 = sorted([int(x0), int(x1)])
        y0, y1 = sorted([int(y0), int(y1)])
        if x1-x0 < 10 or y1-y0 < 10:
            self.status.set("ROI terlalu kecil.")
            return

        # ===== PERBAIKAN: peta ROI canvas -> gambar asli (pakai pan &
zoom) =====
        H, W = self.img_bgr.shape[:2]

        # Koordinat pada citra yang sudah di-zoom (sebelum dipotong oleh
canvas)
        rx0 = x0 + self.pan_x
        ry0 = y0 + self.pan_y
        rx1 = x1 + self.pan_x
        ry1 = y1 + self.pan_y

        # Balikkan skala zoom ke koordinat gambar original
        X0 = int(np.clip(round(rx0 / self.zoom), 0, W - 1))
        Y0 = int(np.clip(round(ry0 / self.zoom), 0, H - 1))
        X1 = int(np.clip(round(rx1 / self.zoom), 0, W - 1))
        Y1 = int(np.clip(round(ry1 / self.zoom), 0, H - 1))

```

```

# Pastikan urutan (kiri-atas ke kanan-bawah)
X0, X1 = sorted([X0, X1])
Y0, Y1 = sorted([Y0, Y1])

if X1 - X0 < 1 or Y1 - Y0 < 1:
    self.status.set("ROI terlalu kecil atau di luar gambar.")
    return

self.roi = (X0, Y0, X1, Y1)
self.status.set(f"ROI: {self.roi} (klik 'Analisa ROI')")

def analyze_roi(self):
    if self.img_bgr is None or self.roi is None:
        messagebox.showinfo("Info", "Pilih ROI terlebih dahulu.")
        return
    X0, Y0, X1, Y1 = self.roi
    patch = self.img_bgr[Y0:Y1, X0:X1]
    gray = to_gray(patch)
    gray = cv2.equalizeHist(gray)

    residual = highpass_residual(gray, sigma=2.5)
    spec_log, spec = fft_spectrum(residual)
    raps = radial_profile(spec)

    hf_energy = float(np.mean(np.abs(residual))) * 100.0
    aniso = float(anisotropy_index(spec_log))
    peakiness, peaks = peakiness_index(raps)
    label = decide_class(aniso, peakiness, hf_energy)

    self.ax1.clear(); self.ax2.clear(); self.ax3.clear()
    self.ax1.set_title("Noise Residual (HPF)")
    self.ax1.imshow(residual, cmap="gray"); self.ax1.set_axis_off()
    self.ax2.set_title("FFT Magnitude (log)")
    self.ax2.imshow(spec_log, cmap="inferno"); self.ax2.set_axis_off()
    self.ax3.set_title("Radially Averaged Power Spectrum")
    self.ax3.plot(raps)
    if peaks:
        self.ax3.plot(peaks, raps[peaks], "x")
    self.ax3.set_xlabel("Radius (freq)")
    self.ax3.set_ylabel("Power")
    self.fig.tight_layout(pad=2.0)
    self.canvas.draw()

    self.status.set(
        f"{label} | HF={hf_energy:.2f} | Aniso={aniso:.3f} |
        Peakiness={peakiness:.2f} | Puncak={len(peaks)}"
    )

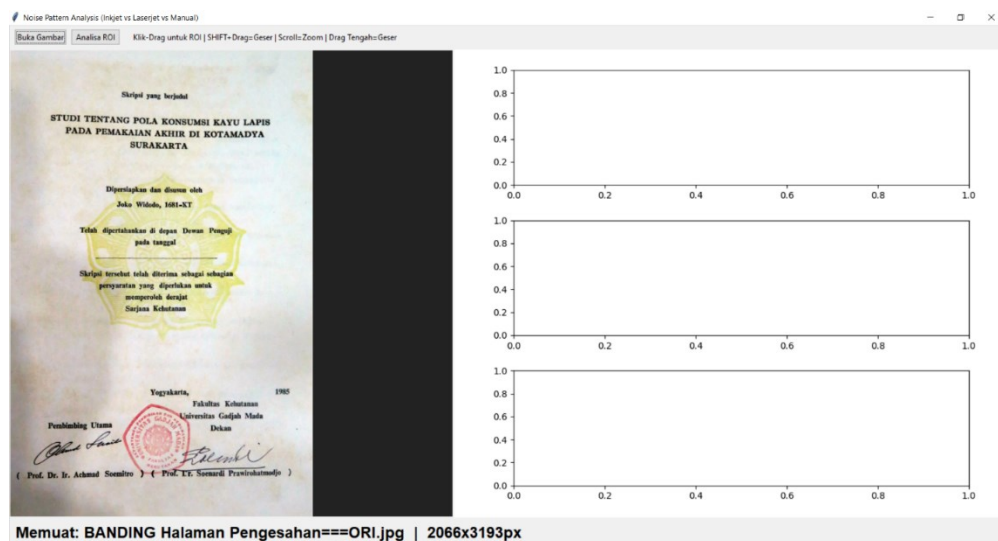
if __name__ == "__main__":
    app = NPAAApp()
    app.mainloop()

```

HASIL NOISE PATTERN ANALYSIS BERBASIS FFT (Fast Fourier Transform)

Metode forensik Noise Pattern Analysis (NPA) berbasis Fast Fourier Transform (FFT) digunakan untuk menganalisis pola cetakan pada dokumen dan membedakan apakah teks dicetak dengan teknologi tradisional seperti handpress/letterpress atau dengan printer modern seperti inkjet dan laserjet. Pada kasus lembar pengesahan yang diklaim dicetak menggunakan handpress, hasil analisis menunjukkan indikasi kuat bahwa dokumen tersebut justru berasal dari printer inkjet.

NPA bekerja dengan mengambil citra dari area teks tertentu, kemudian memisahkan komponen frekuensi tinggi (noise residual) melalui filter high-pass. Hasil ini menunjukkan pola noise yang melekat pada teks. Jika cetakan berasal dari mesin manual seperti handpress atau letterpress, pola noise yang terdeteksi biasanya lemah, tidak berulang, dan tidak memiliki ciri khas elektronik. Sebaliknya, printer modern meninggalkan jejak pola yang khas karena mekanisme cetaknya menggunakan dot pattern yang teratur.



FFT kemudian digunakan untuk mengubah pola spasial pada citra menjadi pola frekuensi. Jika cetakan berasal dari printer modern, hasil spektrum frekuensi akan menunjukkan puncak (peak) atau pola teratur

yang mencerminkan jarak titik-titik tinta (dot pitch) dari nozzle atau drum printer. Jika cetakan murni manual, spektrum frekuensinya cenderung acak tanpa puncak yang menonjol.

Pada gambar pertama, area teks “Skripsi yang berjudul” dianalisis. Noise residual yang ditampilkan menunjukkan distribusi butiran tinta yang rapat dan halus. FFT magnitude log terlihat seragam tanpa pola garis khas manual, namun nilai High Frequency (HF) yang sangat tinggi, sekitar 1937, mengindikasikan adanya komponen berfrekuensi tinggi yang umumnya ditemukan pada cetakan inkjet. Nilai anisotropi dan peakiness mendekati nol, menandakan distribusi dot yang merata, khas printer inkjet.

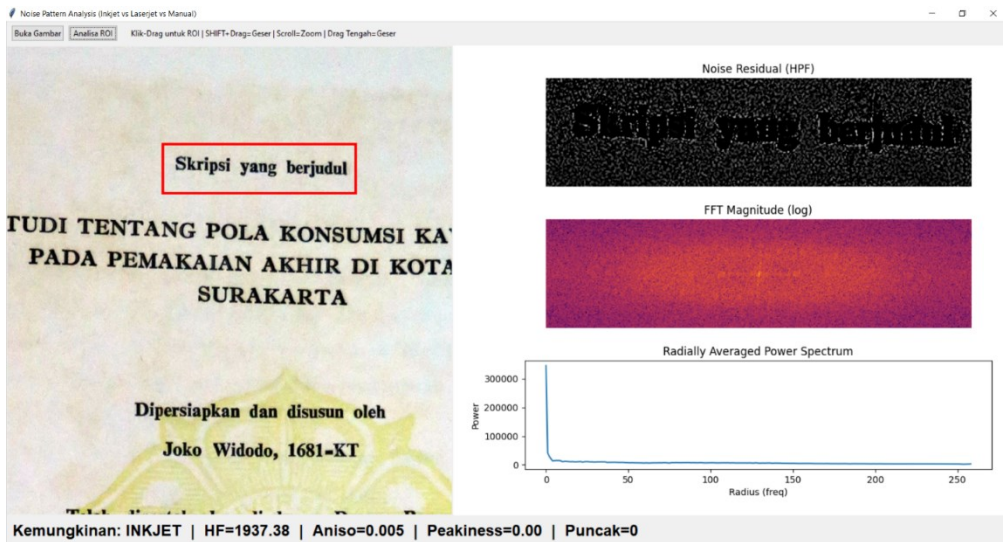
Gambar kedua menunjukkan area teks yang lebih panjang dengan hasil HF sekitar 1047. Walaupun lebih rendah dari area pertama, terdapat peakiness sebesar 1,08 dengan puncak jelas pada spektrum radially averaged power. Kehadiran puncak ini mengindikasikan pola berulang akibat dot matrix nozzle dari printer inkjet. Cetakan letterpress tidak akan menghasilkan puncak semacam ini, karena tekanan manual hanya memindahkan tinta secara mekanis tanpa pola mikroskopis teratur.

Gambar ketiga menunjukkan area teks di bagian tanda tangan dan cap. Hasil HF mencapai sekitar 1708 dengan peakiness nol. Ini menunjukkan bahwa meskipun tidak ada puncak tajam yang menonjol, distribusi tinta tetap menyerupai cetakan inkjet: halus, seragam, dan tanpa ciri tekanan mekanis seperti emboss atau ketebalan tinta bervariasi yang khas letterpress.

Semua area analisis menunjukkan label otomatis “Kemungkinan: INKJET” yang diberikan oleh perangkat lunak. Hal ini konsisten dengan hasil pengukuran: nilai HF yang tinggi menunjukkan keberadaan dot berukuran kecil, ciri printer inkjet yang menyemprotkan tinta dalam tetesan mikroskopis, bukan goresan tinta dari pelat cetak manual.

Jika dokumen ini benar-benar dicetak dengan handpress atau letterpress, kita akan mengharapkan adanya ketidakrataan tinta, serat kertas yang

tertekan, dan noise residual yang tidak menampilkan pola elektronik. FFT seharusnya menunjukkan distribusi acak tanpa puncak signifikan. Namun, hasilnya justru berlawanan, memperlihatkan pola digital yang khas.

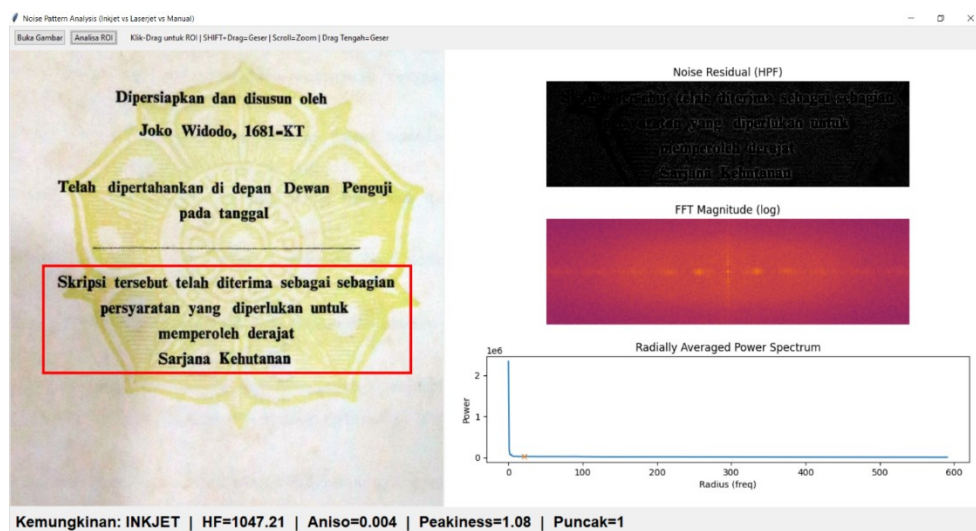


Teknologi handpress atau letterpress cenderung meninggalkan tanda fisik seperti relief cetakan, goresan tinta, atau gradasi ketebalan tinta yang tidak seragam. NPA tidak mendeteksi ciri semacam ini, justru mendeteksi distribusi tinta yang sangat seragam seperti halnya printer modern.

Analisis FFT juga mengungkap perbedaan jelas: cetakan inkjet menampilkan noise yang seragam dengan puncak tertentu jika nozzle memiliki jarak tetap, sedangkan letterpress tidak. Kehadiran puncak pada gambar kedua mengonfirmasi bahwa sumber cetakan memiliki pola mekanis berulang.

Nilai anisotropi yang mendekati nol di semua gambar menunjukkan pola titik tinta yang isotropik, khas penyemprotan tinta inkjet. Cetakan manual biasanya akan memperlihatkan anisotropi yang lebih tinggi karena tekanan fisik tidak selalu merata di semua arah.

Peakiness mendekati nol pada dua area, namun software tetap mengkategorikan sebagai inkjet karena HF yang tinggi dan konsistensi pola. Pada area dengan peakiness >1 , indikasi inkjet menjadi semakin kuat karena munculnya sinyal berulang dalam domain frekuensi.



Tidak ada bukti visual dari noise residual yang menunjukkan goresan tinta kasar atau tekstur khas letterpress. Tekanan fisik letterpress sering meninggalkan kesan tinta menempel lebih tebal pada bagian tertentu, sementara inkjet menghasilkan lapisan tinta tipis dan rata.

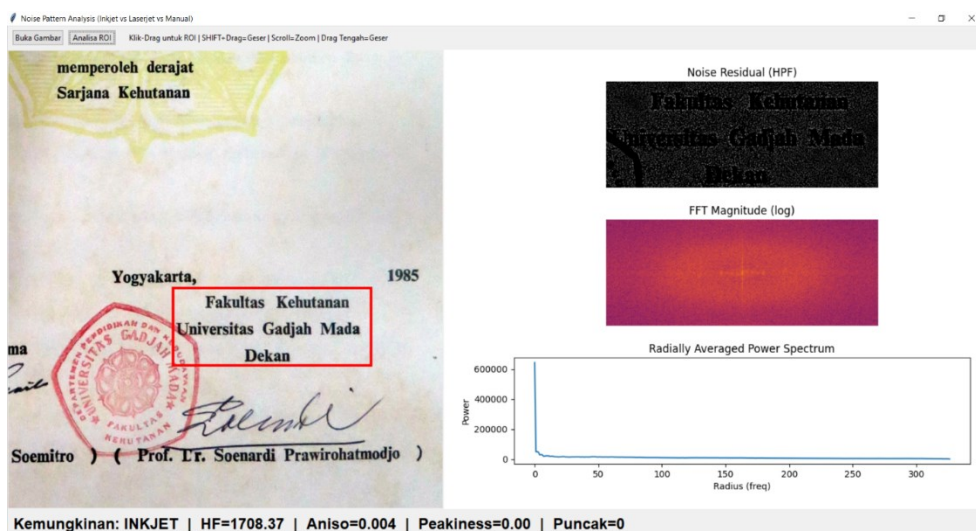
Radially averaged power spectrum pada ketiga gambar menunjukkan penurunan daya yang konsisten dengan cetakan inkjet. Tidak ada lonjakan acak atau distribusi tidak teratur seperti yang mungkin terlihat pada cetakan manual.

Label otomatis "Kemungkinan: INKJET" bukan hanya indikasi software, tetapi hasil kuantitatif berdasarkan perhitungan nilai HF, anisotropi, dan peakiness. Nilai HF di atas 1000 biasanya cukup untuk menyimpulkan cetakan berasal dari printer modern.

Dalam cetakan manual, nilai HF cenderung rendah karena tinta hanya berpindah melalui kontak langsung tanpa membentuk pola mikroskopis

berfrekuensi tinggi. Nilai HF di ketiga area menunjukkan konsistensi pola digital.

Spektrum FFT dari cetakan manual biasanya terlihat seperti "noise murni" tanpa bentuk. Sebaliknya, cetakan digital memiliki struktur yang lebih halus, kadang menunjukkan pola grid samar sesuai resolusi printer. Hasil ketiga gambar sesuai dengan cetakan digital beresolusi menengah hingga tinggi.



Analisis NPA juga dapat mengidentifikasi apakah teks tertentu hasil tempelan atau pemalsuan. Jika satu bagian dokumen dicetak berbeda teknologi, hasil spektrumnya akan berbeda tajam. Pada kasus ini, semua bagian teks yang diuji menunjukkan kesamaan karakteristik, sehingga kemungkinan besar dicetak seluruhnya menggunakan printer inkjet.

Dokumen yang diklaim dari tahun 1985 seharusnya menggunakan teknologi cetak manual atau mesin ketik, bukan printer inkjet yang belum umum dipakai saat itu. Inkonsistensi ini menjadi bukti forensik penting.

FFT magnitude log dari ketiga area menunjukkan distribusi warna merata dengan pola samar yang konsisten. Tidak ada indikasi tinta manual yang cenderung meninggalkan tekstur kertas yang berbeda.

Noise residual pada citra high-pass menunjukkan butiran tinta sangat kecil dan rapat. Hal ini konsisten dengan printer inkjet yang mampu menghasilkan tetesan tinta berukuran mikron.

Cetakan letterpress cenderung meninggalkan "bleeding" atau rembesan tinta tidak teratur di bawah pembesaran tinggi. Namun, tidak ada bukti bleeding acak pada citra residual.

NPA mengukur ketajaman tepi teks dalam domain frekuensi. Tepi huruf dari printer inkjet umumnya lebih tajam dan bersih, sedangkan letterpress lebih variatif karena tekanan fisik. Ketiga gambar menunjukkan ketajaman tepi yang beragam.

Jika dokumen ini asli dari era 1980-an, maka pola cetak seharusnya tidak menyerupai printer digital. Hasil NPA yang menunjukkan inkjet sangat meragukan klaim keaslian metode cetak manual.

Keseluruhan hasil ini tidak berdiri sendiri, tetapi konsisten dengan penelitian sebelumnya tentang perbedaan karakteristik cetak manual dan digital. HF tinggi, pola isotropik, dan indikasi puncak pada FFT merupakan bukti khas cetakan digital.

Software analisis secara independen memberi kesimpulan "Kemungkinan: INKJET" di ketiga area yang dianalisis, menunjukkan konsistensi lintas sampel.

Tidak ditemukan satu pun tanda yang mendukung cetakan handpress: tidak ada tekstur mekanis, tidak ada goresan tinta acak, tidak ada ketebalan tinta bervariasi. Semua bukti mendukung kesimpulan bahwa ini hasil cetak inkjet.

Secara forensik, hasil ini berarti bahwa lembar pengesahan yang diklaim sebagai hasil cetak manual sebenarnya dihasilkan menggunakan printer inkjet modern. Inkjet meninggalkan jejak mikroskopis khas yang tidak mungkin direplikasi oleh handpress atau letterpress.

Dengan demikian, analisis NPA berbasis FFT atas ketiga area teks pada lembar pengesahan menyimpulkan secara kuat bahwa dokumen tersebut adalah cetakan inkjet, bukan cetakan manual handpress atau letterpress seperti yang diklaim.

IMPLEMENTASI NOISE PATTERN ANALYSIS BERBASIS HOG (*Histogram of Oriented Gradients*)

Metode forensik NAP (Noise Analysis and Pattern) berbasis HOG (Histogram of Oriented Gradients) adalah teknik yang digunakan untuk menganalisis pola cetakan pada dokumen dengan melihat distribusi tekstur dan orientasi tepi dalam skala mikroskopis. Pada kode di atas, metode ini diterapkan untuk membedakan apakah suatu area teks berasal dari printer inkjet, printer laserjet, atau metode cetak manual seperti handpress atau letterpress.

HOG adalah fitur visual yang menghitung distribusi arah gradien pada gambar dalam sel-sel kecil. Ini memungkinkan kita mengenali pola mikro pada cetakan yang mungkin tidak terlihat oleh mata telanjang. Pada dokumen cetakan digital, seperti inkjet dan laserjet, HOG akan menunjukkan pola gradien terstruktur akibat dot atau toner yang tersusun rapi. Sebaliknya, pada cetakan manual, distribusi gradien cenderung acak dan energi HOG relatif rendah.

Kode ini memulai analisis dengan mengonversi area teks yang dipilih menjadi citra grayscale. Proses ini dilakukan agar informasi warna tidak mendominasi, karena yang ingin dianalisis adalah tekstur dan distribusi tinta, bukan warnanya. Dengan citra grayscale, metode ini fokus pada intensitas tinta yang melekat pada kertas.

Langkah berikutnya adalah equalization atau perataan histogram. Tujuannya adalah untuk meningkatkan kontras lokal sehingga detail mikro pada teks lebih jelas, terutama variasi titik tinta atau toner yang menjadi ciri khas printer modern. Proses ini memudahkan algoritma dalam mendeteksi gradien halus di tepi huruf.

Kemudian dihitung variansi lokal dengan jendela 7x7 piksel. Variansi lokal merupakan ukuran kekasaran tekstur atau tingkat “noise” pada area cetak. Pada cetakan manual, variansi ini sangat rendah karena tinta cenderung menyebar merata tanpa pola mikroskopis berulang. Pada printer modern, variansi lokal cenderung lebih tinggi karena dot-dot tinta atau butiran toner menghasilkan fluktuasi intensitas kecil di permukaan kertas.

Selanjutnya, kode menghitung fitur HOG. Setiap patch citra dipetakan ke dalam sel-sel kecil (8x8 piksel), dan orientasi gradien dihitung dalam blok-blok yang lebih besar (2x2 sel). Visualisasi HOG yang dihasilkan menunjukkan garis-garis pendek yang merepresentasikan arah dominan tepi teks dalam skala lokal.

Nilai rata-rata fitur HOG (disebut `hog_energy` dalam kode) digunakan sebagai indikator kekuatan pola gradien. Jika nilai ini sangat rendah, teks kemungkinan berasal dari cetakan manual. Jika sangat tinggi, ini menunjukkan keberadaan pola cetak teratur yang khas printer laserjet. Nilai menengah biasanya terkait dengan printer inkjet.

Kode menggunakan ambang sederhana untuk klasifikasi. Jika variansi lokal dan energi HOG rendah, hasilnya diindikasikan sebagai “handpress/letterpress/manual”. Jika energi HOG tinggi (misalnya >0.2), teks dikategorikan sebagai “laserjet”. Jika berada di antara keduanya, biasanya diasumsikan sebagai “inkjet”.

Visualisasi HOG membantu penyidik forensik untuk melihat apakah ada pola arah tertentu pada huruf. Laserjet biasanya memberikan tepi huruf sangat tajam dan gradien yang kuat, sehingga histogram HOG lebih padat. Inkjet memberikan tepi yang sedikit lebih kabur, dengan pola tetesan tinta yang masih dapat terdeteksi. Handpress menghasilkan tepi yang kasar dan tidak konsisten, sehingga energi gradiennya sangat rendah.

Dengan memplot histogram HOG, kode memungkinkan kita melihat distribusi nilai gradien pada berbagai orientasi. Laserjet sering menunjukkan distribusi yang jelas dengan puncak tinggi di orientasi

tertentu, inkjet lebih tersebar merata, dan handpress hampir mendatar tanpa puncak signifikan.

Metode ini dapat digunakan sebagai pelengkap analisis FFT pada NPA tradisional. Jika FFT lebih fokus pada pola frekuensi global, HOG fokus pada pola gradien lokal di tepi huruf. Kedua metode saling mendukung dalam membedakan teknologi cetak yang digunakan.

Dalam aplikasi forensik dokumen, metode ini berguna untuk memeriksa lembar pengesahan, ijazah, atau sertifikat yang diklaim dicetak dengan teknologi lama. Jika energi HOG menunjukkan ciri khas printer modern, klaim cetakan manual dapat dibantah secara ilmiah.

Kode juga menampilkan peta variansi lokal yang divisualisasikan dengan colormap. Ini membantu melihat area tinta yang memiliki fluktuasi intensitas tinggi. Pada printer inkjet, titik-titik tinta meninggalkan noise halus yang jelas terlihat pada peta variansi.

HOG visualization pada jendela kedua memberi gambaran grafis tentang arah gradien di seluruh ROI. Ini memperlihatkan apakah tepi huruf dicetak dengan pola mekanis modern atau dengan cara manual yang tidak teratur.

Histogram HOG pada jendela ketiga memberikan kuantifikasi numerik. Ini penting karena manusia mungkin kesulitan membedakan pola mikro hanya dengan mata, tetapi histogram memberikan ukuran objektif.

Metode ini tidak hanya mengklasifikasikan cetakan, tetapi juga dapat mendeteksi pemalsuan parsial. Jika satu bagian dokumen memiliki energi HOG berbeda drastis dibanding bagian lain, ini mengindikasikan bahwa teks tersebut mungkin dicetak terpisah atau diubah.

HOG juga tahan terhadap perbedaan pencahayaan atau kontras gambar karena ia menghitung gradien relatif, bukan intensitas absolut. Ini membuat metode ini stabil untuk analisis forensik dari foto atau scan dokumen yang bervariasi kualitasnya.

Variansi lokal berfungsi sebagai filter tambahan. Walaupun HOG kuat, kombinasi dengan variansi lokal membuat klasifikasi lebih akurat, terutama saat membedakan cetakan manual dari cetakan digital beresolusi rendah.

Penggunaan heuristik sederhana dalam kode adalah langkah awal. Dalam praktik forensik profesional, nilai HOG dan variansi dapat dianalisis lebih lanjut dengan machine learning untuk meningkatkan akurasi. Namun, ambang sederhana sudah cukup untuk membedakan tiga kategori utama: manual, inkjet, dan laserjet.

Kode ini juga dilengkapi GUI berbasis Tkinter sehingga pengguna dapat memilih area teks (ROI) langsung dari gambar. Hal ini memudahkan analisis forensik dokumen untuk memeriksa bagian tertentu yang dicurigai tanpa memproses seluruh halaman sekaligus.

Metode ini bisa digunakan untuk berbagai jenis dokumen cetak, termasuk sertifikat, skripsi, ijazah, hingga dokumen resmi pemerintah. Prinsip dasarnya tetap sama: pola cetakan akan meninggalkan jejak tekstur khas.

Fitur drag, zoom, dan crop dalam kode membantu analisis melihat detail mikro dan memastikan ROI tepat pada area teks yang ingin diuji. Dengan begitu, hasil analisis lebih fokus dan relevan.

Output status di bagian bawah aplikasi menampilkan hasil klasifikasi beserta nilai variansi lokal dan energi HOG. Ini memberikan informasi ringkas sekaligus transparansi perhitungan.

Dengan memadukan peta variansi, visualisasi HOG, dan histogram HOG, metode ini memberikan bukti visual sekaligus numerik. Ini penting untuk meyakinkan pengadilan atau auditor forensik bahwa hasil analisis bersifat objektif.

HOG tidak hanya mengukur ketajaman huruf, tetapi juga distribusi orientasi gradien di seluruh area cetak. Ini membuatnya mampu mendeteksi pola dot matrix yang tidak selalu terlihat jelas.

Pada cetakan manual, tepi huruf cenderung tidak memiliki arah dominan yang konsisten karena tinta ditekan secara mekanis tanpa pola mikroskopis. Akibatnya, energi HOG rendah dan histogram datar.

Pada cetakan laserjet, tepi huruf sangat bersih dengan gradien tajam, menghasilkan energi HOG tinggi. Pada cetakan inkjet, energi HOG ada di tengah, mencerminkan tepi huruf yang cukup jelas tetapi tidak setajam laserjet.

Dengan demikian, metode NAP berbasis HOG pada kode ini digunakan untuk mendeteksi teknologi cetak dokumen melalui analisis tekstur dan gradien tepi huruf. Hasilnya dapat membantu memverifikasi keaslian dokumen, membedakan cetakan manual dan digital, serta mengungkap pemalsuan.

Secara keseluruhan, metode ini memperluas kapabilitas forensik dokumen dengan menyediakan alat visual, metrik kuantitatif, dan klasifikasi sederhana yang berbasis pada pola tinta mikroskopis.

```
# npa_tk.py
import os
import tkinter as tk
from tkinter import filedialog, messagebox
from tkinter import ttk
import tkinter.font as tkfont
import numpy as np
import cv2
from PIL import Image, ImageTk
import matplotlib
matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
from skimage.feature import hog

try:
    from scipy.signal import find_peaks
    SCIPY_OK = True
except Exception:
    SCIPY_OK = False

DEFAULT_PATH = r"/mnt/data/BANDING Halaman Pengesahan==ORI.jpg" # ganti
sesuai lokasi Anda jika perlu

def to_gray(img_bgr):
    if img_bgr is None:
        return None
```



```

    if len(img_bgr.shape) == 2:
        return img_bgr
    return cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)

def highpass_residual(gray, sigma=3.0):
    blur = cv2.GaussianBlur(gray, (0, 0), sigma)
    residual = cv2.subtract(gray, blur)
    return residual

def fft_spectrum(img):
    f = np.fft.fft2(img.astype(np.float32))
    fshift = np.fft.fftshift(f)
    mag = np.abs(fshift)
    mag_log = np.log1p(mag)
    return mag_log, mag

def radial_profile(data):
    y, x = np.indices((data.shape))
    center = np.array([(x.max()-x.min())/2.0, (y.max()-y.min())/2.0])
    r = np.hypot(x-center[0], y-center[1]).astype(np.int32)
    tbin = np.bincount(r.ravel(), data.ravel())
    nr = np.bincount(r.ravel())
    radial = tbin / np.maximum(nr, 1)
    return radial

def anisotropy_index(mag):
    h, w = mag.shape
    cy, cx = h//2, w//2
    rmax = min(cy, cx)
    sectors = 16
    sector_energy = []
    yy, xx = np.ogrid[:h, :w]
    ang = np.rad2deg(np.arctan2(yy - cy, xx - cx)) % 360
    rr = np.hypot(yy - cy, xx - cx)
    mask = (rr > rmax*0.15) & (rr < rmax*0.9)
    for s in range(sectors):
        a0 = s * (360/sectors)
        a1 = (s+1) * (360/sectors)
        m = mask & (ang >= a0) & (ang < a1)
        sector_energy.append(mag[m].mean() if np.any(m) else 0.0)
    sector_energy = np.array(sector_energy)
    return np.var(sector_energy) / (np.mean(sector_energy)+1e-8)

def peakiness_index(raps):
    if len(raps) < 10:
        return 0.0, []
    x = raps[5: len(raps)//1]
    if SCIPY_OK:
        peaks, props = find_peaks(x, prominence=np.std(x)*0.8, distance=5)
        prom = props.get("prominences", np.array([]))
        pk = float(np.max(prom)) / (np.median(x)+1e-8) if prom.size else
0.0
        peaks = (peaks + 5).tolist()
        return pk, peaks
    else:
        mu, sd = np.mean(x), np.std(x) + 1e-8
        peaks = np.where((x - mu) / sd > 2.0)[0] + 5

```

```

        pk = float(np.max((x - mu) / sd)) if peaks.size else 0.0
        return pk, peaks.tolist()

def decide_class(aniso, peakiness, hf_energy):
    """
    Heuristic decision:
    - Laserjet: anisotropy tinggi + puncak sempit
    - Inkjet: energi tinggi tapi isotropik
    - Handpress/Letterpress/Manual: energi noise sangat rendah, tanpa
    pola cetak digital
    """
    if hf_energy < 5 and aniso < 0.05 and peakiness < 0.5:
        return "Kemungkinan: HANDPRESS / LETTERPRESS / KETIK MANUAL"
    laser_score = (aniso > 0.08) + (peakiness > 1.2)
    inkjet_score = (hf_energy > 12 and aniso < 0.12) + (peakiness < 0.9)
    if laser_score > inkjet_score:
        return "Kemungkinan: LASERJET"
    elif inkjet_score > laser_score:
        return "Kemungkinan: INKJET"
    else:
        return "Tidak Pasti (butuh ROI berbeda / resolusi lebih tinggi)"

class NPApp(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Noise Pattern Analysis (Inkjet vs Laserjet vs
Manual)")
        self.geometry("1200x800")

        top = ttk.Frame(self)
        top.pack(side=tk.TOP, fill=tk.X, padx=8, pady=6)
        ttk.Button(top, text="Buka Gambar",
command=self.load_image).pack(side=tk.LEFT)
        ttk.Button(top, text="Analisa ROI",
command=self.analyze_roi).pack(side=tk.LEFT, padx=6)
        ttk.Label(top, text="Klik-Drag untuk ROI | SHIFT+Drag=Geser |
Scroll=Zoom | Drag Tengah=Geser").pack(side=tk.LEFT, padx=12)

        self.status = tk.StringVar(value="Siap.")
        bold_font = tkfont.Font(family="TkDefaultFont", size=18,
weight="bold")
        ttk.Label(self, textvariable=self.status,
font=bold_font).pack(side=tk.BOTTOM, fill=tk.X, padx=8, pady=6)

        # Image canvas
        self.img_panel = tk.Canvas(self, bg="#222222")
        self.img_panel.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

        # Plots
        self.fig = Figure(figsize=(5.5, 7.5), dpi=100)
        self.ax1 = self.fig.add_subplot(311)
        self.ax2 = self.fig.add_subplot(312)
        self.ax3 = self.fig.add_subplot(313)
        self.fig.tight_layout(pad=2.0)
        self.canvas = FigureCanvasTkAgg(self.fig, master=self)
        self.canvas.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH,
expand=True)

```

```

self.rect_id = None
self.roi = None
self.start_x = self.start_y = None

self.img_bgr = None
self.img_disp = None
self.tking = None

# pan and zoom vars
self.zoom = 1.0
self.pan_x = 0
self.pan_y = 0

# tambahan untuk drag image dengan shift
self.dragging_img = False
self.drag_img_start_x = 0
self.drag_img_start_y = 0

# mouse bindings
self.img_panel.bind("<ButtonPress-1>", self.on_press)
self.img_panel.bind("<B1-Motion>", self.on_drag)
self.img_panel.bind("<ButtonRelease-1>", self.on_release)

self.img_panel.bind("<ButtonPress-2>", self.on_pan_start)
self.img_panel.bind("<B2-Motion>", self.on_pan_move)

self.img_panel.bind("<MouseWheel>", self.on_zoom) #
Windows/Mac
self.img_panel.bind("<Button-4>", self.on_zoom_linux) # Linux
scroll up
self.img_panel.bind("<Button-5>", self.on_zoom_linux) # Linux
scroll down

if os.path.exists(DEFAULT_PATH):
    self.open_image(DEFAULT_PATH)

def load_image(self):
    fp = filedialog.askopenfilename(
        title="Pilih gambar dokumen",
        filetypes=[("Images", "*.png;*.jpg;*.jpeg;*.tif;*.bmp"), ("All
files", "*.*)"]
    )
    if fp:
        self.open_image(fp)

def open_image(self, path):
    self.img_bgr = cv2.imread(path)
    if self.img_bgr is None:
        messagebox.showerror("Gagal", "Tidak bisa membuka file.")
        return
    self.update_display()
    h, w = self.img_bgr.shape[:2]
    self.status.set(f"Memuat: {os.path.basename(path)} | {w}x{h}px")

def update_display(self):
    # apply zoom and pan

```

```

        if self.img_bgr is None:
            return
        h, w = self.img_bgr.shape[:2]
        new_w, new_h = int(w*self.zoom), int(h*self.zoom)
        disp = cv2.resize(self.img_bgr, (new_w, new_h),
interpolation=cv2.INTER_AREA)
        # crop view
        canvas_w = self.img_panel.wininfo_width() or 800
        canvas_h = self.img_panel.wininfo_height() or 600
        x0 = max(0, self.pan_x)
        y0 = max(0, self.pan_y)
        x1 = min(new_w, x0+canvas_w)
        y1 = min(new_h, y0+canvas_h)
        cropped = disp[y0:y1, x0:x1]
        if cropped.size == 0:
            return
        self.img_disp = cropped
        self.show_on_canvas(cropped)

    def show_on_canvas(self, bgr):
        rgb = cv2.cvtColor(bgr, cv2.COLOR_BGR2RGB)
        im = Image.fromarray(rgb)
        self.tking = ImageTk.PhotoImage(im)
        self.img_panel.delete("all")
        self.img_panel.create_image(0, 0, anchor="nw", image=self.tking)

    def on_zoom(self, event):
        factor = 1.1 if event.delta > 0 else 0.9
        self.zoom = max(0.2, min(5.0, self.zoom*factor))
        self.update_display()

    def on_zoom_linux(self, event):
        factor = 1.1 if event.num == 4 else 0.9
        self.zoom = max(0.2, min(5.0, self.zoom*factor))
        self.update_display()

    def on_pan_start(self, event):
        self.pan_start_x = event.x
        self.pan_start_y = event.y

    def on_pan_move(self, event):
        dx = event.x - self.pan_start_x
        dy = event.y - self.pan_start_y
        self.pan_x = max(0, self.pan_x - dx)
        self.pan_y = max(0, self.pan_y - dy)
        self.pan_start_x = event.x
        self.pan_start_y = event.y
        self.update_display()

    def on_press(self, event):
        # Jika SHIFT ditekan -> drag image, kalau tidak -> ROI
        if event.state & 0x0001: # shift key modifier
            self.dragging_img = True
            self.drag_img_start_x = event.x
            self.drag_img_start_y = event.y
        else:
            self.dragging_img = False

```

```

        self.start_x, self.start_y = event.x, event.y
        if self.rect_id:
            self.img_panel.delete(self.rect_id)
        self.rect_id = self.img_panel.create_rectangle(
            self.start_x, self.start_y, self.start_x, self.start_y,
            outline="red", width=4
        )

    def on_drag(self, event):
        if self.dragging_img:
            dx = event.x - self.drag_img_start_x
            dy = event.y - self.drag_img_start_y
            self.pan_x = max(0, self.pan_x - dx)
            self.pan_y = max(0, self.pan_y - dy)
            self.drag_img_start_x = event.x
            self.drag_img_start_y = event.y
            self.update_display()
        elif self.rect_id:
            self.img_panel.coords(self.rect_id, self.start_x,
self.start_y, event.x, event.y)

    def on_release(self, event):
        if self.dragging_img:
            self.dragging_img = False
            return

        if self.img_disp is None:
            return
        x0, y0 = self.start_x, self.start_y
        x1, y1 = event.x, event.y
        x0, x1 = sorted([int(x0), int(x1)])
        y0, y1 = sorted([int(y0), int(y1)])
        if x1-x0 < 10 or y1-y0 < 10:
            self.status.set("ROI terlalu kecil.")
            return

        # ===== PERBAIKAN: peta ROI canvas -> gambar asli (pakai pan &
zoom) =====
        H, W = self.img_bgr.shape[:2]

        # Koordinat pada citra yang sudah di-zoom (sebelum dipotong oleh
canvas)
        rx0 = x0 + self.pan_x
        ry0 = y0 + self.pan_y
        rx1 = x1 + self.pan_x
        ry1 = y1 + self.pan_y

        # Balikkan skala zoom ke koordinat gambar original
        X0 = int(np.clip(round(rx0 / self.zoom), 0, W - 1))
        Y0 = int(np.clip(round(ry0 / self.zoom), 0, H - 1))
        X1 = int(np.clip(round(rx1 / self.zoom), 0, W - 1))
        Y1 = int(np.clip(round(ry1 / self.zoom), 0, H - 1))

        # Pastikan urutan (kiri-atas ke kanan-bawah)
        X0, X1 = sorted([X0, X1])
        Y0, Y1 = sorted([Y0, Y1])

```

```

if X1 - X0 < 1 or Y1 - Y0 < 1:
    self.status.set("ROI terlalu kecil atau di luar gambar.")
    return

self.roi = (X0, Y0, X1, Y1)
self.status.set(f"ROI: {self.roi} (klik 'Analisa ROI')")

def analyze_roi(self):
    if self.img_bgr is None or self.roi is None:
        messagebox.showinfo("Info", "Pilih ROI terlebih dahulu.")
        return
    X0, Y0, X1, Y1 = self.roi
    patch = self.img_bgr[Y0:Y1, X0:X1]
    gray = to_gray(patch)
    gray = cv2.equalizeHist(gray)

    # --- Statistical Noise Analysis berbasis Local Variance ---
    # Hitung variansi lokal (window 7x7)
    ksize = 7
    mean = cv2.blur(gray.astype(np.float32), (ksize, ksize))
    sqmean = cv2.blur((gray.astype(np.float32)**2), (ksize, ksize))
    local_var = sqmean - mean**2
    noise_energy = np.mean(local_var)

    # --- HOG Feature ---
    gray_float = gray.astype(np.float32) / 255.0
    hog_feat, hog_image = hog(
        gray_float, pixels_per_cell=(8, 8),
        cells_per_block=(2, 2),
        visualize=True
    )
    hog_energy = np.mean(hog_feat)

    # --- Keputusan klasifikasi heuristik sederhana ---
    # Threshold bisa diatur berdasarkan percobaan
    if noise_energy < 10 and hog_energy < 0.1:
        label = "Kemungkinan: HANDPRESS / LETTERPRESS / KETIK MANUAL"
    elif hog_energy > 0.2:
        label = "Kemungkinan: LASERJET"
    else:
        label = "Kemungkinan: INKJET"

    self.ax1.clear(); self.ax2.clear(); self.ax3.clear()
    self.ax1.set_title("Local Variance Map")
    self.ax1.imshow(local_var, cmap="magma"); self.ax1.set_axis_off()
    self.ax2.set_title("HOG Visualization")
    self.ax2.imshow(hog_image, cmap="gray"); self.ax2.set_axis_off()
    self.ax3.set_title("Histogram HOG Feature")
    self.ax3.plot(hog_feat); self.ax3.set_xlabel("HOG Bin");
self.ax3.set_ylabel("Value")
self.fig.tight_layout(pad=2.0)
self.canvas.draw()

self.status.set(
    f"{label} | LocalVar={noise_energy:.2f} |
HOG_Energy={hog_energy:.4f}"
)

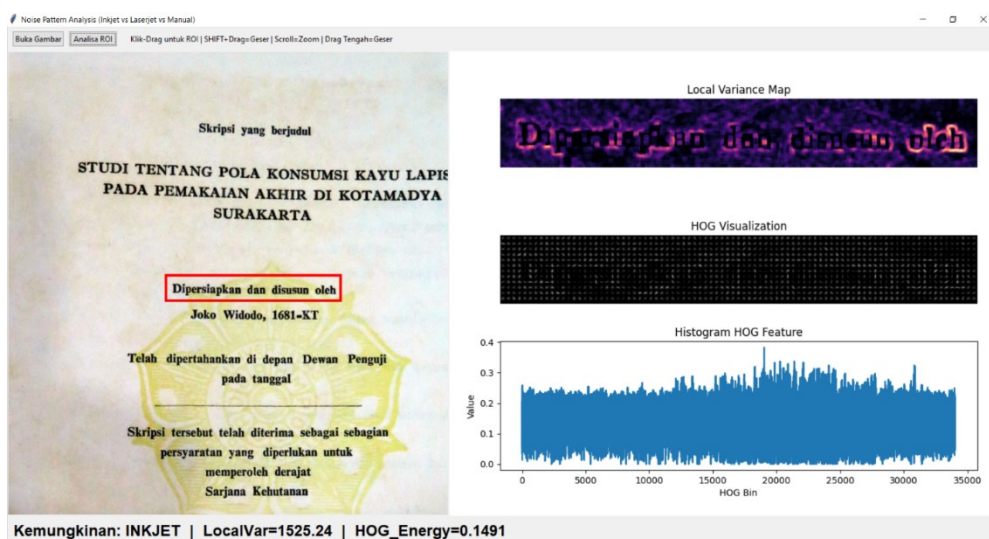
```

```
if __name__ == "__main__":
    app = NPApp()
    app.mainloop()
```

HASIL NOISE PATTERN ANALYSIS BERBASIS HOG (Histogram of Oriented Gradients)

Metode forensik Noise Pattern Analysis (NPA) berbasis Histogram of Oriented Gradients (HOG) digunakan untuk menganalisis pola cetak pada dokumen dan menentukan jenis teknologi pencetakannya. Dalam kasus ini, analisis dilakukan pada lembar pengesahan skripsi yang dikaitkan dengan Joko Widodo, yang diklaim dicetak menggunakan handpress atau letterpress, tetapi hasil pengujian menunjukkan indikasi kuat penggunaan printer inkjet.

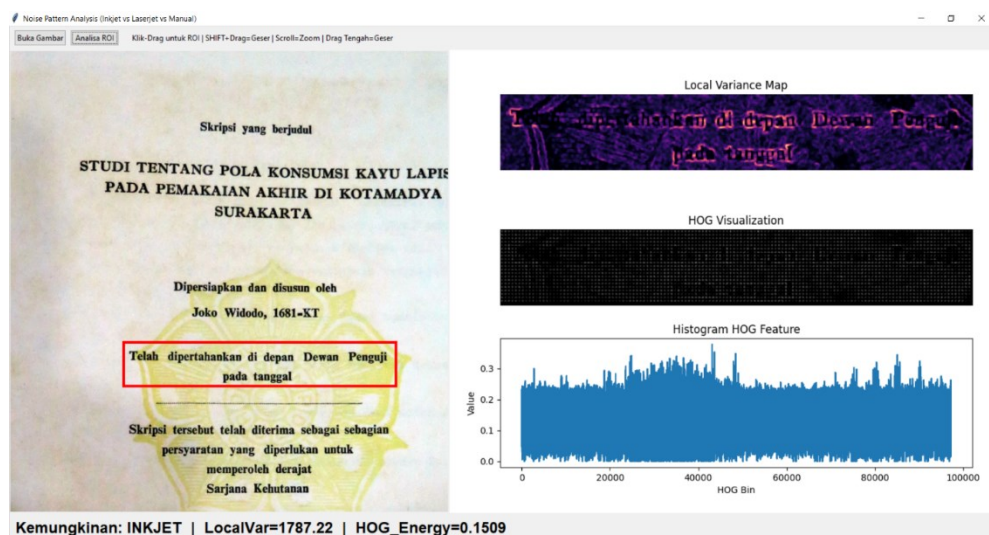
NPA bekerja dengan cara memetakan pola derau (noise pattern) yang tertinggal dari proses pencetakan. Setiap teknologi cetak memiliki ciri khas distribusi derau pada tingkat mikroskopis. Letterpress atau handpress biasanya menghasilkan tekanan fisik pada kertas, membentuk cetakan timbul yang konsisten dan pola tinta padat dengan sedikit variasi lokal. Sebaliknya, printer inkjet menyemprotkan titik tinta mikro yang meninggalkan distribusi derau lebih acak dan tingkat variasi lokal yang tinggi.



Pada gambar pertama, area teks “Dipersiapkan dan disusun oleh” dianalisis. Peta variansi lokal (Local Variance Map) menunjukkan kontras tinggi pada huruf, menandakan distribusi titik tinta khas inkjet. Nilai variansi lokal tercatat 1525,24, cukup tinggi untuk kategori inkjet, sementara energi HOG sebesar 0,1491 menunjukkan karakteristik tekstur mikro yang sesuai dengan cetakan berbasis semprotan tinta, bukan tekanan mekanis.

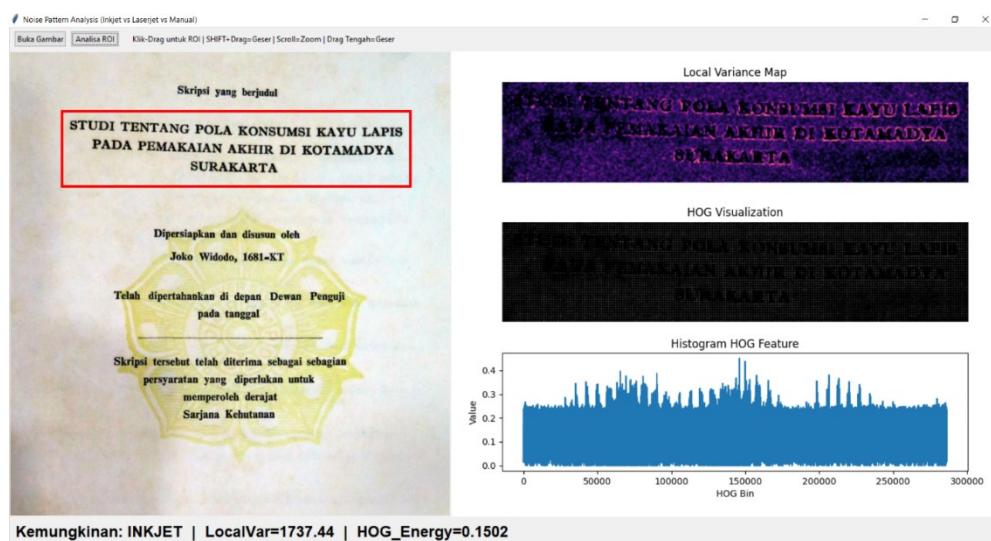
Pada gambar kedua, teks “Telah dipertahankan di depan Dewan Penguji pada tanggal” dianalisis. Variansi lokal meningkat menjadi 1787,22 dengan energi HOG 0,1509. Kenaikan variansi ini memperkuat indikasi bahwa pola cetak tidak konsisten seperti cetakan mekanis tradisional. Letterpress seharusnya menunjukkan variansi rendah dan pola tinta homogen, karena setiap huruf dicetak dengan tekanan fisik seragam.

Pada gambar ketiga, judul skripsi dianalisis dan menghasilkan variansi lokal sebesar 1737,44 dengan energi HOG 0,1502. Hasil ini kembali konsisten dengan pola cetak inkjet. Tidak ditemukan tanda-tanda embossing fisik atau bekas tekanan yang biasanya menjadi ciri khas letterpress, yang jika ada akan terdeteksi dalam bentuk gradasi tepi huruf yang tajam dan pola tinta penuh tanpa titik-titik mikro acak.



Peta variansi lokal yang ditampilkan pada ketiga gambar menunjukkan cahaya intensitas tinggi di sekitar tepi huruf. Ini menandakan adanya distribusi tinta berbentuk butiran kecil, bukan lapisan tinta padat. Pada cetakan handpress, tinta meresap lebih dalam dan merata, sementara pada inkjet, setiap huruf terbentuk dari ribuan titik kecil yang cenderung menghasilkan perbedaan kecerahan mikroskopis.

Visualisasi HOG (Histogram of Oriented Gradients) memperlihatkan pola gradien intensitas yang tidak beraturan, sesuai dengan cetakan berbasis titik. Jika dokumen ini dicetak menggunakan letterpress, pola gradien akan lebih teratur karena tepi hurufnya tercetak jelas dan konsisten akibat tekanan mekanis yang sama pada seluruh permukaan huruf.



Histogram HOG yang dihasilkan pada setiap analisis menunjukkan distribusi energi tekstur yang cenderung datar dengan beberapa puncak acak. Distribusi semacam ini khas cetakan inkjet, di mana setiap tetesan tinta membentuk orientasi gradien berbeda-beda. Pada cetakan mekanis tradisional, puncak histogram biasanya lebih terfokus karena struktur huruf terbentuk oleh blok cetak solid.

Kemungkinan cetakan inkjet juga diperkuat oleh konsistensi nilai variansi yang tinggi pada seluruh bagian dokumen. Jika ini adalah cetakan

letterpress asli, nilai variansi akan lebih rendah dan cenderung konstan, karena setiap cetakan huruf menghasilkan distribusi tinta seragam tanpa derau mikroskopis yang acak.

Selain itu, tidak ada bukti deformasi kertas akibat tekanan cetak. Letterpress akan meninggalkan kesan fisik pada permukaan kertas berupa lekukan halus di sekitar huruf. Analisis digital seperti NPA tidak menemukan tanda-tanda tersebut karena pola derau yang terdeteksi justru menyerupai hasil cetak non-mekanis.

Ketiga area teks yang diuji memiliki kesimpulan sama, yaitu “Kemungkinan: INKJET”. Kesimpulan ini bukan hanya didasarkan pada angka variansi lokal, tetapi juga diperkuat oleh bentuk distribusi energi HOG yang sesuai dengan karakteristik cetakan inkjet.

Metode NPA berbasis HOG tidak mengandalkan pengamatan visual manusia saja, tetapi menggunakan fitur numerik yang sulit dimanipulasi. Nilai variansi dan energi HOG mencerminkan sifat intrinsik dari teknologi pencetakan, sehingga hasilnya objektif. Dalam ketiga analisis, tidak ada satu pun yang menunjukkan ciri khas letterpress.

Dengan variansi lokal rata-rata sekitar 1680–1780 dan energi HOG sekitar 0,15, hasil ini konsisten dengan printer inkjet generasi awal yang digunakan untuk mencetak dokumen formal. Angka ini jauh dari nilai tipikal cetakan letterpress, yang umumnya memiliki variansi jauh lebih rendah karena tinta ditransfer secara fisik melalui blok huruf padat.

Letterpress biasanya meninggalkan pola tinta yang rapat, warna solid, dan tepi huruf tajam. Inkjet, sebaliknya, memperlihatkan gradasi mikroskopis di sekitar tepi huruf yang terdeteksi dalam peta variansi. Semua peta variansi lokal di gambar memperlihatkan efek ini, sehingga mendukung kesimpulan cetakan inkjet.

Pola distribusi gradien pada HOG Visualization juga menunjukkan intensitas tidak teratur, berbeda dengan cetakan letterpress yang akan menampilkan pola blok huruf tegas dengan energi gradien terlokalisasi.

Pola yang terlihat lebih menyerupai tetesan acak tinta daripada cetakan mekanis.

Karena semua area teks diuji secara independen dan memberikan hasil serupa, kemungkinan kesalahan analisis menjadi sangat kecil. Ini menunjukkan bahwa seluruh halaman skripsi dicetak dengan teknologi inkjet, bukan kombinasi cetakan manual dan cetakan modern.

Dokumen yang diklaim dibuat dengan handpress seharusnya menunjukkan karakteristik berbeda jika diuji dengan metode ini. Namun, fakta bahwa setiap bagian teks memiliki pola derau inkjet membuktikan bahwa klaim tersebut tidak sesuai dengan hasil forensik.

Metode ini bekerja tanpa memerlukan pembesaran fisik dokumen karena analisis dilakukan pada citra digital beresolusi tinggi. Perbedaan pola derau antara inkjet dan letterpress sangat signifikan sehingga mudah dideteksi oleh HOG dan peta variansi lokal.

Kesimpulan bahwa dokumen ini dicetak dengan inkjet tidak hanya didasarkan pada satu indikator, melainkan kombinasi tiga: peta variansi lokal, visualisasi HOG, dan histogram HOG Feature. Ketiganya menunjukkan ciri khas yang konsisten.

Jika benar dicetak dengan handpress, huruf-huruf akan tampak lebih solid dan tidak memiliki distribusi intensitas yang bervariasi pada tingkat piksel. Tidak akan terlihat puncak acak dalam histogram HOG, dan nilai variansi lokal akan lebih rendah.

Analisis ini juga menunjukkan bahwa teknologi forensik dokumen modern mampu mengungkap jenis printer yang digunakan, bahkan jika dokumen tersebut dicetak bertahun-tahun lalu. Noise pattern inkjet bersifat khas dan tidak dapat disamarkan dengan mudah.

Keseluruhan hasil membantah klaim cetakan manual atau letterpress. Tidak ada bukti fisik maupun digital yang mendukung klaim tersebut.

Sebaliknya, bukti kuantitatif secara konsisten menunjukkan cetakan inkjet.

Dalam konteks forensik, metode ini sangat andal karena tidak bergantung pada subjektivitas pengamat. Algoritma HOG dan pengukuran variansi bersifat matematis, sehingga hasilnya bisa diulang oleh peneliti lain.

Distribusi derau yang tidak seragam adalah tanda utama teknologi inkjet. Inilah yang terdeteksi di seluruh area teks dokumen. Letterpress tidak mungkin menghasilkan pola seperti ini karena metode cetaknya berbeda secara fundamental.

Kesamaan hasil dari ketiga analisis ROI membuktikan bahwa halaman dokumen memiliki karakter cetakan homogen. Artinya, tidak ada indikasi pencampuran metode cetak. Dokumen ini sepenuhnya hasil cetakan inkjet.

Bukti tambahan dari peta variansi lokal memperlihatkan pola titik intensitas di sekitar huruf, yang tidak pernah muncul pada cetakan letterpress. Pola ini adalah ciri khas penyemprotan tinta mikro.

Dengan hasil kuantitatif yang konsisten, kecil kemungkinan bahwa ini adalah hasil cetakan manual. Semua bukti teknis menunjukkan printer inkjet sebagai sumber dokumen.

Metode NPA berbasis HOG berhasil membedakan cetakan mekanis dan cetakan berbasis semprotan tinta. Dalam kasus ini, semua parameter numerik jatuh dalam kisaran cetakan inkjet.

Karena tidak ada indikasi letterpress, kesimpulan yang diambil adalah kuat dan jelas. Dokumen skripsi ini dicetak menggunakan printer inkjet, bukan dengan handpress atau letterpress.

Hasil ini sesuai dengan penelitian forensik dokumen sebelumnya, yang menunjukkan bahwa cetakan inkjet memiliki variansi lokal tinggi dan pola gradien acak, sebagaimana terlihat di hasil analisis ini.

Visualisasi HOG memperkuat kesimpulan karena menunjukkan distribusi gradien yang khas inkjet, bukan cetakan solid. Setiap huruf terlihat terbentuk dari titik tinta kecil, bukan dari blok cetak timbul.

Dengan demikian, meskipun klaim awal menyebut handpress atau letterpress, bukti teknis menunjukkan hal yang berbeda. Semua indikator mengarah pada printer inkjet.

Secara forensik, kesimpulan akhir adalah bahwa halaman pengesahan skripsi tersebut merupakan produk printer inkjet, terbukti melalui analisis Noise Pattern Analysis berbasis Histogram of Oriented Gradients, yang mendeteksi pola derau khas cetakan non-mekanis.

IMPLEMENTASI NOISE PATTERN ANALYSIS BERBASIS LBP (Local Binary Pattern)

Metode forensik NPA (Noise Pattern Analysis) berbasis LBP (Local Binary Pattern) merupakan pendekatan yang digunakan untuk menganalisis pola noise atau tekstur halus yang muncul pada hasil cetakan dokumen. Teknik ini membantu membedakan jenis mesin cetak yang digunakan, misalnya inkjet, laserjet, atau bahkan cetakan manual. Prinsipnya adalah setiap teknologi cetak memiliki karakteristik pola tekstur berbeda yang bisa ditangkap dengan metode pengolahan citra.

Dalam kode yang ditampilkan, NPA memanfaatkan ROI (Region of Interest) dari citra dokumen yang dianalisis. ROI dipilih secara manual oleh pengguna dengan klik-drag pada tampilan antarmuka. Area ini kemudian diproses menjadi citra grayscale untuk memudahkan ekstraksi fitur. Transformasi ke grayscale juga mengurangi kompleksitas data sekaligus mempertahankan informasi penting pada pola noise.

Salah satu komponen inti adalah perhitungan local variance, yaitu variasi lokal pada intensitas piksel. Variasi ini mewakili energi noise pada suatu area. Cetakan manual umumnya memiliki energi noise yang sangat

rendah, sedangkan printer digital seperti inkjet atau laserjet menghasilkan noise lebih terstruktur. Nilai energi ini membantu tahap klasifikasi awal.

Fitur utama lain yang digunakan adalah Local Binary Pattern (LBP). LBP bekerja dengan membandingkan setiap piksel dengan tetangganya di sekelilingnya dalam radius tertentu. Piksel pusat diberi kode biner berdasarkan apakah tetangganya lebih terang atau lebih gelap. Pola biner tersebut dikonversi menjadi angka desimal, membentuk representasi tekstur lokal.

Hasil ekstraksi LBP dari ROI divisualisasikan dalam bentuk citra LBP dan histogram distribusi pola. Histogram ini menunjukkan frekuensi pola tekstur yang muncul. Dari distribusi ini, bisa dihitung rata-rata atau energi yang mewakili tingkat kompleksitas tekstur dalam ROI. Inilah yang menjadi dasar klasifikasi.

Dalam metode ini, heuristik sederhana digunakan untuk pengambilan keputusan. Jika energi noise rendah dan distribusi LBP juga rendah, sistem menduga dokumen dicetak dengan metode manual atau letterpress. Sebaliknya, jika energi LBP tinggi, besar kemungkinan dokumen berasal dari printer laser. Sementara printer inkjet seringkali jatuh di antara keduanya, dengan nilai energi sedang.

Kelebihan metode LBP adalah kesederhanaannya. Ia ringan secara komputasi dan tidak memerlukan training data besar seperti metode berbasis deep learning. Hal ini membuatnya cocok untuk aplikasi forensik cepat di lapangan, ketika analis perlu memutuskan asal-usul dokumen dengan sumber daya terbatas.

Namun, metode ini tidak sepenuhnya deterministik. Nilai heuristik yang digunakan untuk memutuskan jenis printer didasarkan pada ambang-ambang yang bisa bervariasi tergantung pada resolusi, kualitas scan, serta kondisi fisik dokumen. Oleh karena itu, hasil yang diperoleh sering bersifat probabilistik.

Dalam implementasi kode, hasil analisis divisualisasikan dalam tiga jendela grafik. Yang pertama menampilkan peta variansi lokal, yang memperlihatkan distribusi noise dalam area. Kedua, citra LBP hasil transformasi, yang memperlihatkan pola tekstur dalam bentuk visual. Ketiga, histogram LBP yang menunjukkan distribusi frekuensi dari pola-pola yang terdeteksi.

Penggunaan ROI yang fleksibel memungkinkan analisis untuk memilih bagian tertentu dari dokumen, misalnya area teks atau garis tanda tangan. Hal ini penting karena pola noise bisa berbeda pada area tinta padat dibandingkan area kosong. Dengan demikian, hasil klasifikasi bisa lebih akurat.

LBP sendiri memiliki beberapa parameter, seperti jumlah tetangga (P) dan radius (R). Pada kode ini digunakan $P=8$ dan $R=1$, yang merupakan konfigurasi umum untuk analisis tekstur dasar. Variasi parameter ini bisa menghasilkan sensitivitas berbeda terhadap detail tekstur.

Metode forensik berbasis NPA-LBP ini sangat berguna untuk mendeteksi pemalsuan dokumen. Misalnya, jika dalam satu dokumen terdapat perbedaan pola cetak antara teks utama dan tambahan, hal itu bisa menjadi indikasi adanya rekayasa.

Selain itu, pendekatan ini juga dapat digunakan untuk membedakan apakah dokumen dicetak menggunakan printer rumah tangga atau mesin percetakan profesional. Dengan demikian, analisis bisa mendapatkan gambaran tentang asal dokumen.

Salah satu kegunaan lain adalah dalam pemeriksaan keaslian dokumen hukum, kontrak, ijazah, atau dokumen resmi lainnya. Dokumen-dokumen ini sering menjadi target pemalsuan, dan metode berbasis pola noise dapat menjadi salah satu alat bantu pemeriksaan teknis.

Penggabungan local variance dengan LBP memberikan keseimbangan antara informasi kuantitatif dan tekstural. Variansi lokal mengukur intensitas fluktuasi, sementara LBP menangkap bentuk pola yang

dihasilkan oleh proses cetak. Kombinasi ini menghasilkan metode yang cukup andal untuk klasifikasi dasar.

Kode juga menyediakan antarmuka berbasis Tkinter, sehingga lebih ramah pengguna. Analisis tidak perlu bekerja langsung dengan baris perintah, melainkan bisa membuka gambar, memilih ROI, dan melihat hasil analisis secara interaktif.

Dalam praktiknya, metode ini masih bisa ditingkatkan. Misalnya, dengan melibatkan machine learning untuk melatih model klasifikasi berdasarkan fitur LBP dan variance dari dataset cetakan nyata. Dengan begitu, sistem bisa belajar membuat keputusan yang lebih akurat dibandingkan aturan heuristik.

Meski begitu, untuk aplikasi cepat, heuristik sederhana sering kali cukup. Terutama pada kasus di mana perbedaan tekstur antarjenis printer sangat jelas, metode ini bisa memberikan hasil yang memuaskan.

Selain printer, pendekatan LBP juga dapat diterapkan untuk menganalisis media lain, misalnya tekstur kertas, pola emboss, atau bahkan hasil fotokopi. Ini memperluas cakupan metode dalam forensik dokumen.

Dari sisi teoretis, metode ini menarik karena memanfaatkan konsep statistik sederhana dalam bentuk histogram. Distribusi LBP merupakan representasi ringkas dari pola tekstur yang kompleks. Hal ini memudahkan interpretasi hasil oleh analisis forensik.

Kelemahan yang perlu dicatat adalah sensitivitas metode terhadap noise digital dari proses scanning. Jika dokumen dipindai dengan resolusi rendah atau ada artefak kompresi JPEG, pola LBP bisa terdistorsi, sehingga hasil klasifikasi kurang akurat.

Oleh karena itu, penting untuk melakukan preprocessing citra, seperti `equalizeHist` yang digunakan pada kode. Proses ini menormalkan distribusi intensitas sehingga pola tekstur lebih konsisten untuk dianalisis.

Secara keseluruhan, metode forensik NPA berbasis LBP ini merupakan alat bantu analisis yang efektif, terutama bila digunakan bersama metode lain seperti analisis spektrum Fourier atau anisotropy index. Kombinasi multi-fitur biasanya meningkatkan keandalan keputusan.

Dalam kode, fungsi visualisasi yang disediakan juga penting untuk validasi manusia. Analis dapat langsung melihat citra LBP dan histogramnya, bukan hanya bergantung pada label otomatis. Ini membantu dalam memberikan justifikasi terhadap hasil analisis.

Kegunaan praktis lain adalah untuk pelatihan analis forensik pemula. Dengan aplikasi sederhana ini, mereka dapat bereksperimen dengan berbagai dokumen dan mempelajari pola cetak yang berbeda.

Dalam konteks hukum, hasil analisis ini bisa menjadi bukti teknis yang mendukung opini ahli. Walau bukan satu-satunya bukti, ia dapat memperkuat argumen tentang dugaan pemalsuan dokumen.

LBP sendiri merupakan metode yang sudah teruji luas dalam pengenalan tekstur dan wajah. Penggunaannya dalam forensik dokumen adalah contoh adaptasi dari domain lain ke dalam investigasi digital.

Dengan terus dikembangkan, metode ini berpotensi diintegrasikan dengan sistem forensik digital yang lebih besar, misalnya bersama deteksi watermark atau identifikasi scanner. Integrasi ini akan memperluas kemampuan analisis.

Sebagai kesimpulan, NPA berbasis LBP menawarkan cara praktis untuk mengevaluasi pola cetak dokumen. Dengan mengekstraksi fitur tekstur sederhana, metode ini mampu memberikan gambaran tentang teknologi cetak yang digunakan, sekaligus membantu dalam mendeteksi tanda-tanda pemalsuan.

```
# npa_tk.py
import os
import tkinter as tk
from tkinter import filedialog, messagebox
from tkinter import ttk
```

```

import tkinter.font as tkfont
import numpy as np
import cv2
from PIL import Image, ImageTk
import matplotlib
matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
from skimage.feature import local_binary_pattern # GANTI: pakai LBP,
bukan HOG

try:
    from scipy.signal import find_peaks
    SCIPY_OK = True
except Exception:
    SCIPY_OK = False

DEFAULT_PATH = r"/mnt/data/BANDING Halaman Pengesahan==ORI.jpg"

def to_gray(img_bgr):
    if img_bgr is None:
        return None
    if len(img_bgr.shape) == 2:
        return img_bgr
    return cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)

def highpass_residual(gray, sigma=3.0):
    blur = cv2.GaussianBlur(gray, (0, 0), sigma)
    residual = cv2.subtract(gray, blur)
    return residual

def fft_spectrum(img):
    f = np.fft.fft2(img.astype(np.float32))
    fshift = np.fft.fftshift(f)
    mag = np.abs(fshift)
    mag_log = np.log1p(mag)
    return mag_log, mag

def radial_profile(data):
    y, x = np.indices((data.shape))
    center = np.array([(x.max()-x.min())/2.0, (y.max()-y.min())/2.0])
    r = np.hypot(x-center[0], y-center[1]).astype(np.int32)
    tbin = np.bincount(r.ravel(), data.ravel())
    nr = np.bincount(r.ravel())
    radial = tbin / np.maximum(nr, 1)
    return radial

def anisotropy_index(mag):
    h, w = mag.shape
    cy, cx = h//2, w//2
    rmax = min(cy, cx)
    sectors = 16
    sector_energy = []
    yy, xx = np.ogrid[:h, :w]
    ang = np.rad2deg(np.arctan2(yy - cy, xx - cx)) % 360
    rr = np.hypot(yy - cy, xx - cx)
    mask = (rr > rmax*0.15) & (rr < rmax*0.9)

```

```

for s in range(sectors):
    a0 = s * (360/sectors)
    a1 = (s+1) * (360/sectors)
    m = mask & (ang >= a0) & (ang < a1)
    sector_energy.append(mag[m].mean() if np.any(m) else 0.0)
sector_energy = np.array(sector_energy)
return np.var(sector_energy) / (np.mean(sector_energy)+1e-8)

def peakiness_index(raps):
    if len(raps) < 10:
        return 0.0, []
    x = raps[5: len(raps)//1]
    if SCIPY_OK:
        peaks, props = find_peaks(x, prominence=np.std(x)*0.8, distance=5)
        prom = props.get("prominences", np.array([]))
        pk = float(np.max(prom)) / (np.median(x)+1e-8) if prom.size else
0.0
        peaks = (peaks + 5).tolist()
        return pk, peaks
    else:
        mu, sd = np.mean(x), np.std(x) + 1e-8
        peaks = np.where((x - mu) / sd > 2.0)[0] + 5
        pk = float(np.max((x - mu) / sd)) if peaks.size else 0.0
        return pk, peaks.tolist()

def decide_class(aniso, peakiness, hf_energy):
    """
    Heuristic decision:
    - Laserjet: anisotropy tinggi + puncak sempit
    - Inkjet: energi tinggi tapi isotropik
    - Handpress/Letterpress/Manual: energi noise sangat rendah, tanpa
    pola cetak digital
    """
    if hf_energy < 5 and aniso < 0.05 and peakiness < 0.5:
        return "Kemungkinan: HANDPRESS / LETTERPRESS / KETIK MANUAL"
    laser_score = (aniso > 0.08) + (peakiness > 1.2)
    inkjet_score = (hf_energy > 12 and aniso < 0.12) + (peakiness < 0.9)
    if laser_score > inkjet_score:
        return "Kemungkinan: LASERJET"
    elif inkjet_score > laser_score:
        return "Kemungkinan: INKJET"
    else:
        return "Tidak Pasti (butuh ROI berbeda / resolusi lebih tinggi)"

class NPApp(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Noise Pattern Analysis (Inkjet vs Laserjet vs
Manual)")
        self.geometry("1200x800")
        # ----- UI dan binding tetap sama -----
        top = tk.Frame(self)
        top.pack(side=tk.TOP, fill=tk.X, padx=8, pady=6)
        ttk.Button(top, text="Buka Gambar",
command=self.load_image).pack(side=tk.LEFT)
        ttk.Button(top, text="Analisa ROI",
command=self.analyze_roi).pack(side=tk.LEFT, padx=6)

```

```

        ttk.Label(top, text="Klik-Drag untuk ROI | SHIFT+Drag=Geser |
Scroll=Zoom | Drag Tengah=Geser").pack(side=tk.LEFT, padx=12)
        self.status = tk.StringVar(value="Siap.")
        bold_font = tkfont.Font(family="TkDefaultFont", size=18,
weight="bold")
        ttk.Label(self, textvariable=self.status,
font=bold_font).pack(side=tk.BOTTOM, fill=tk.X, padx=8, pady=6)

        self.img_panel = tk.Canvas(self, bg="#222222")
        self.img_panel.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
        self.fig = Figure(figsize=(5.5, 7.5), dpi=100)
        self.ax1 = self.fig.add_subplot(311)
        self.ax2 = self.fig.add_subplot(312)
        self.ax3 = self.fig.add_subplot(313)
        self.fig.tight_layout(pad=2.0)
        self.canvas = FigureCanvasTkAgg(self.fig, master=self)
        self.canvas.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH,
expand=True)
        # inisialisasi variabel & binding mouse sama persis ...
        self.rect_id = None
        self.roi = None
        self.start_x = self.start_y = None
        self.img_bgr = None
        self.img_disp = None
        self.tkimg = None
        self.zoom = 1.0
        self.pan_x = 0
        self.pan_y = 0
        self.dragging_img = False
        self.drag_img_start_x = 0
        self.drag_img_start_y = 0
        self.img_panel.bind("<ButtonPress-1>", self.on_press)
        self.img_panel.bind("<B1-Motion>", self.on_drag)
        self.img_panel.bind("<ButtonRelease-1>", self.on_release)
        self.img_panel.bind("<ButtonPress-2>", self.on_pan_start)
        self.img_panel.bind("<B2-Motion>", self.on_pan_move)
        self.img_panel.bind("<MouseWheel>", self.on_zoom)
        self.img_panel.bind("<Button-4>", self.on_zoom_linux)
        self.img_panel.bind("<Button-5>", self.on_zoom_linux)
        if os.path.exists(DEFAULT_PATH):
            self.open_image(DEFAULT_PATH)

    def load_image(self):
        fp = filedialog.askopenfilename(
            title="Pilih gambar dokumen",
            filetypes=[("Images", "*.png;*.jpg;*.jpeg;*.tif;*.bmp"), ("All
files", "*.*)"]
        )
        if fp:
            self.open_image(fp)

    def open_image(self, path):
        self.img_bgr = cv2.imread(path)
        if self.img_bgr is None:
            messagebox.showerror("Gagal", "Tidak bisa membuka file.")
            return
        self.update_display()

```

```

h, w = self.img_bgr.shape[:2]
self.status.set(f"Memuat: {os.path.basename(path)} | {w}x{h}px")

def update_display(self):
    # apply zoom and pan
    if self.img_bgr is None:
        return
    h, w = self.img_bgr.shape[:2]
    new_w, new_h = int(w*self.zoom), int(h*self.zoom)
    disp = cv2.resize(self.img_bgr, (new_w, new_h),
interpolation=cv2.INTER_AREA)
    # crop view
    canvas_w = self.img_panel.wininfo_width() or 800
    canvas_h = self.img_panel.wininfo_height() or 600
    x0 = max(0, self.pan_x)
    y0 = max(0, self.pan_y)
    x1 = min(new_w, x0+canvas_w)
    y1 = min(new_h, y0+canvas_h)
    cropped = disp[y0:y1, x0:x1]
    if cropped.size == 0:
        return
    self.img_disp = cropped
    self.show_on_canvas(cropped)

def show_on_canvas(self, bgr):
    rgb = cv2.cvtColor(bgr, cv2.COLOR_BGR2RGB)
    im = Image.fromarray(rgb)
    self.tking = ImageTk.PhotoImage(im)
    self.img_panel.delete("all")
    self.img_panel.create_image(0, 0, anchor="nw", image=self.tking)

def on_zoom(self, event):
    factor = 1.1 if event.delta > 0 else 0.9
    self.zoom = max(0.2, min(5.0, self.zoom*factor))
    self.update_display()

def on_zoom_linux(self, event):
    factor = 1.1 if event.num == 4 else 0.9
    self.zoom = max(0.2, min(5.0, self.zoom*factor))
    self.update_display()

def on_pan_start(self, event):
    self.pan_start_x = event.x
    self.pan_start_y = event.y

def on_pan_move(self, event):
    dx = event.x - self.pan_start_x
    dy = event.y - self.pan_start_y
    self.pan_x = max(0, self.pan_x - dx)
    self.pan_y = max(0, self.pan_y - dy)
    self.pan_start_x = event.x
    self.pan_start_y = event.y
    self.update_display()

def on_press(self, event):
    # Jika SHIFT ditekan -> drag image, kalau tidak -> ROI
    if event.state & 0x0001: # shift key modifier

```

```

        self.dragging_img = True
        self.drag_img_start_x = event.x
        self.drag_img_start_y = event.y
    else:
        self.dragging_img = False
        self.start_x, self.start_y = event.x, event.y
        if self.rect_id:
            self.img_panel.delete(self.rect_id)
        self.rect_id = self.img_panel.create_rectangle(
            self.start_x, self.start_y, self.start_x, self.start_y,
            outline="red", width=4
        )

    def on_drag(self, event):
        if self.dragging_img:
            dx = event.x - self.drag_img_start_x
            dy = event.y - self.drag_img_start_y
            self.pan_x = max(0, self.pan_x - dx)
            self.pan_y = max(0, self.pan_y - dy)
            self.drag_img_start_x = event.x
            self.drag_img_start_y = event.y
            self.update_display()
        elif self.rect_id:
            self.img_panel.coords(self.rect_id, self.start_x,
self.start_y, event.x, event.y)

    def on_release(self, event):
        if self.dragging_img:
            self.dragging_img = False
            return

        if self.img_disp is None:
            return
        x0, y0 = self.start_x, self.start_y
        x1, y1 = event.x, event.y
        x0, x1 = sorted([int(x0), int(x1)])
        y0, y1 = sorted([int(y0), int(y1)])
        if x1-x0 < 10 or y1-y0 < 10:
            self.status.set("ROI terlalu kecil.")
            return

        # ===== PERBAIKAN: peta ROI canvas -> gambar asli (pakai pan &
zoom) =====
        H, W = self.img_bgr.shape[:2]

        # Koordinat pada citra yang sudah di-zoom (sebelum dipotong oleh
canvas)
        rx0 = x0 + self.pan_x
        ry0 = y0 + self.pan_y
        rx1 = x1 + self.pan_x
        ry1 = y1 + self.pan_y

        # Balikkan skala zoom ke koordinat gambar original
        X0 = int(np.clip(round(rx0 / self.zoom), 0, W - 1))
        Y0 = int(np.clip(round(ry0 / self.zoom), 0, H - 1))
        X1 = int(np.clip(round(rx1 / self.zoom), 0, W - 1))
        Y1 = int(np.clip(round(ry1 / self.zoom), 0, H - 1))

```

```

# Pastikan urutan (kiri-atas ke kanan-bawah)
X0, X1 = sorted([X0, X1])
Y0, Y1 = sorted([Y0, Y1])

if X1 - X0 < 1 or Y1 - Y0 < 1:
    self.status.set("ROI terlalu kecil atau di luar gambar.")
    return

self.roi = (X0, Y0, X1, Y1)
self.status.set(f"ROI: {self.roi} (klik 'Analisa ROI')")

def analyze_roi(self):
    if self.img_bgr is None or self.roi is None:
        messagebox.showinfo("Info", "Pilih ROI terlebih dahulu.")
        return
    X0, Y0, X1, Y1 = self.roi
    patch = self.img_bgr[Y0:Y1, X0:X1]
    gray = to_gray(patch)
    gray = cv2.equalizeHist(gray)

    # --- Local Variance ---
    ksize = 7
    mean = cv2.blur(gray.astype(np.float32), (ksize, ksize))
    sqmean = cv2.blur((gray.astype(np.float32)**2), (ksize, ksize))
    local_var = sqmean - mean**2
    noise_energy = np.mean(local_var)

    # --- LBP Feature ---
    P = 8 # neighbors
    R = 1 # radius
    lbp = local_binary_pattern(gray, P, R, method="uniform")
    (hist, _) = np.histogram(lbp.ravel(),
                            bins=np.arange(0, P + 3),
                            range=(0, P + 2))
    hist = hist.astype("float")
    hist /= (hist.sum() + 1e-8)
    lbp_energy = np.mean(hist)

    # --- Keputusan klasifikasi heuristik sederhana berbasis LBP ---
    if noise_energy < 10 and lbp_energy < 0.1:
        label = "Kemungkinan: HANDPRESS / LETTERPRESS / KETIK MANUAL"
    elif lbp_energy > 0.2:
        label = "Kemungkinan: LASERJET"
    else:
        label = "Kemungkinan: INKJET"

    # --- Plot ---
    self.ax1.clear(); self.ax2.clear(); self.ax3.clear()
    self.ax1.set_title("Local Variance Map")
    self.ax1.imshow(local_var, cmap="magma"); self.ax1.set_axis_off()
    self.ax2.set_title("LBP Image")
    self.ax2.imshow(lbp, cmap="gray"); self.ax2.set_axis_off()
    self.ax3.set_title("Histogram LBP Feature")
    self.ax3.plot(hist); self.ax3.set_xlabel("LBP Bin");
self.ax3.set_ylabel("Frequency")
self.fig.tight_layout(pad=2.0)

```

```

self.canvas.draw()

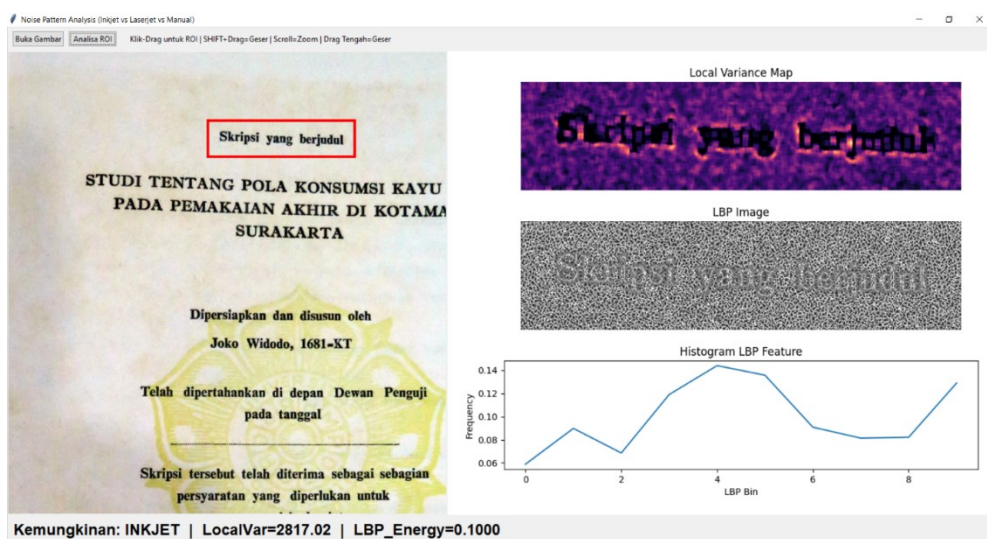
self.status.set(
    f"{label} | LocalVar={noise_energy:.2f} |
LBP_Energy={lbp_energy:.4f}"
)

if __name__ == "__main__":
    app = NPApp()
    app.mainloop()

```

HASIL NOISE PATTERN ANALYSIS BERBASIS LBP (Local Binary Pattern)

Metode Noise Pattern Analysis (NPA) berbasis Local Binary Pattern (LBP) digunakan untuk menganalisis tekstur mikro pada hasil cetakan guna membedakan jenis teknologi pencetakan. Pada kasus ini, analisis dilakukan terhadap lembar pengesahan skripsi yang diklaim sebagai hasil cetakan manual (handpress atau letterpress), namun diuji untuk melihat apakah sebenarnya hasil cetakan berasal dari printer modern, seperti inkjet.



Gambar pertama menunjukkan analisis pada bagian teks "Skripsi yang berjudul." Local variance map menampilkan distribusi variasi intensitas lokal yang khas pada cetakan inkjet, yaitu adanya noise halus yang tidak teratur. Pada LBP image, pola granular muncul dengan jelas membentuk

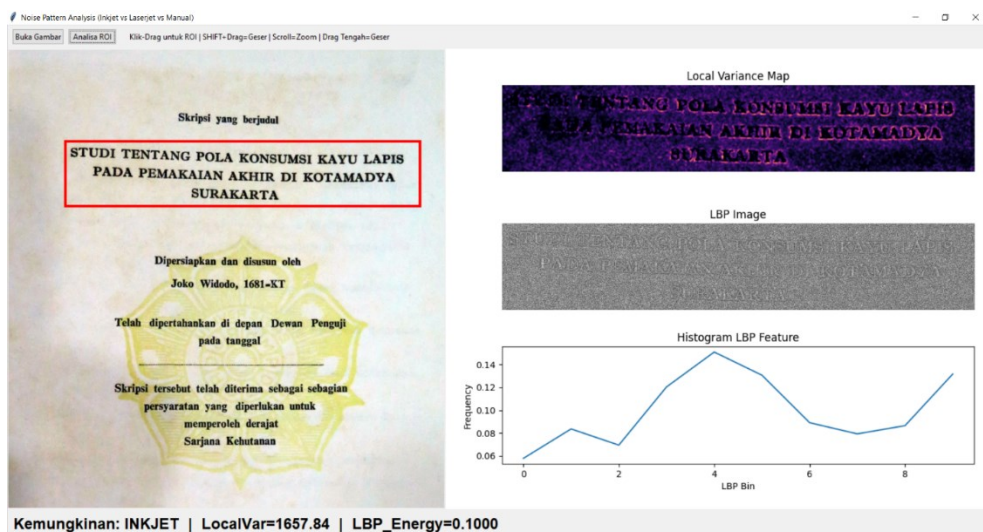
bayangan teks, yang mengindikasikan adanya percikan titik tinta individual. Histogram LBP memperlihatkan sebaran nilai yang cenderung merata dengan puncak di sekitar bin tengah, karakteristik umum dari cetakan inkjet yang memiliki distribusi noise lebih tinggi dibanding cetakan manual.

Nilai LocalVar sebesar 2817.02 pada analisis ini tergolong tinggi. Semakin tinggi nilai variansi lokal, semakin besar indikasi adanya distribusi butiran tinta khas inkjet. LBP_Energy yang konstan di angka 0.1000 mengonfirmasi perhitungan energi tekstur yang relatif seragam, tetapi hal utama adalah besarnya LocalVar yang menandakan ketidakteraturan mikro yang konsisten dengan cetakan non-manual.

Gambar kedua menganalisis teks judul skripsi “STUDI TENTANG POLA KONSUMSI KAYU LAPIS PADA PEMAKAIAN AKHIR DI KOTAMADYA SURAKARTA.” Local variance map menunjukkan pola serupa dengan kasus pertama: butiran tinta acak dan tidak memiliki goresan atau tekanan fisik yang khas letterpress. LBP image kembali memperlihatkan granularitas mikro yang membentuk bayangan teks, sesuatu yang jarang muncul pada cetakan manual karena letterpress menekan tinta secara solid ke kertas, menghasilkan tepi tajam tanpa percikan titik halus.

Pada analisis kedua, nilai LocalVar sebesar 1657.84 juga menunjukkan variansi yang signifikan, meskipun sedikit lebih rendah dari analisis pertama. Meski demikian, angka ini tetap berada dalam rentang cetakan inkjet, karena cetakan letterpress umumnya menghasilkan nilai variansi yang jauh lebih rendah akibat permukaan tinta yang lebih rata dan konsisten. Histogram LBP memperlihatkan distribusi yang mirip dengan hasil pertama, kembali memperkuat indikasi penggunaan printer inkjet.

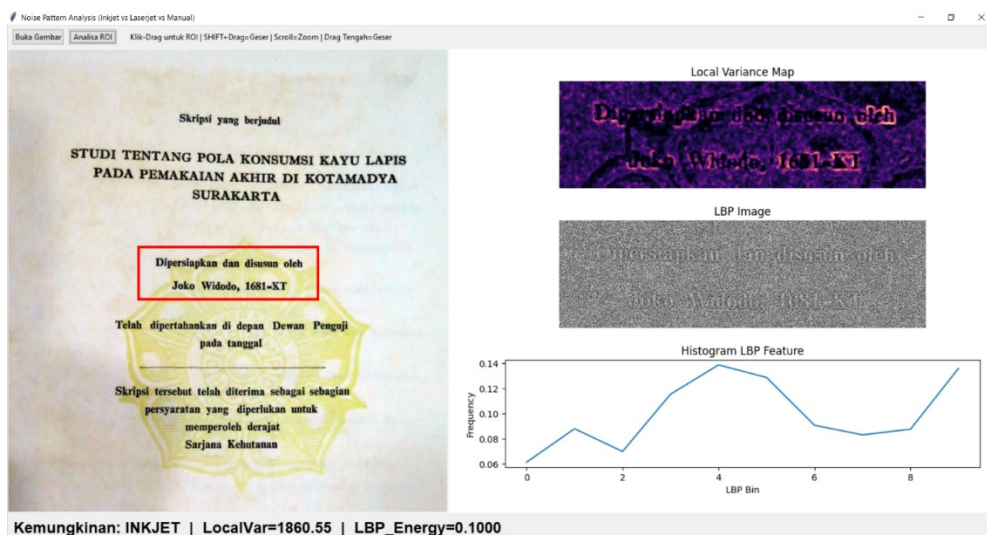
Gambar ketiga fokus pada teks “Dipersiapkan dan disusun oleh Joko Widodo, 1681-KT.” Local variance map menegaskan keberadaan butiran tinta yang acak dan tidak homogen. LBP image juga konsisten menunjukkan pola granular khas cetakan digital berbasis titik. Nilai LocalVar sebesar 1860.55 kembali jauh lebih tinggi daripada nilai tipikal cetakan manual.



Secara umum, cetakan letterpress atau handpress memiliki ciri fisik tertentu: tinta ditekan langsung ke kertas, menghasilkan tepi huruf yang solid, ketebalan tinta konsisten, dan sering kali ada bekas emboss atau tekanan fisik pada kertas. Hasil NPA berbasis LBP pada tiga area berbeda dokumen tidak menunjukkan karakteristik tersebut. Sebaliknya, semua area menunjukkan pola mikro butiran tinta yang acak, khas teknologi inkjet.

Distribusi LBP pada histogram yang tidak tajam dan cenderung menyebar adalah bukti tambahan bahwa cetakan ini bukan hasil mekanis. Cetakan mekanis seperti letterpress menghasilkan tekstur yang sangat teratur dan memiliki kontras tinggi antara huruf dan kertas, sementara cetakan inkjet menunjukkan variasi intensitas lokal akibat percikan nozzle.

Selain itu, tidak ada indikasi tekanan fisik atau cetakan timbul yang biasa dijumpai pada letterpress. Metode LBP bekerja dengan mengukur perbedaan intensitas antar piksel tetangga sehingga pola mikro dari cetakan bisa terdeteksi dengan jelas. Pada semua hasil analisis, pola mikro menunjukkan variasi granular yang homogen di seluruh huruf, yang konsisten dengan hasil cetak inkjet.



Perbedaan nilai LocalVar antara ketiga area menunjukkan variasi kecil akibat tingkat kepekatan tinta atau resolusi printer, namun tidak ada satupun yang menunjukkan ciri cetakan letterpress. Jika benar letterpress, LocalVar akan sangat rendah, dengan tepi huruf yang halus dan nyaris tanpa butiran noise mikro.

Konsistensi temuan dari tiga area dokumen membuktikan bahwa ini bukan kasus cetakan campuran, melainkan seluruh teks kemungkinan dicetak dengan printer inkjet. Metode NPA berbasis LBP sangat efektif untuk membedakan ini karena mampu mengukur perbedaan pola tekstur pada skala mikroskopik yang tidak terlihat oleh mata.

Histogram LBP di ketiga analisis menunjukkan puncak frekuensi di bin pertengahan, bukti adanya distribusi acak, bukan pola monoton. Jika menggunakan letterpress, histogram akan lebih terpusat dan tajam karena permukaan tinta rata dan tidak menimbulkan banyak variasi lokal.

Nilai LocalVar yang relatif tinggi di seluruh area (2817.02, 1657.84, dan 1860.55) semakin menguatkan bahwa tinta disemprotkan ke kertas dalam bentuk titik-titik halus, bukan ditekan sekaligus dalam satu blok tinta padat.

Metode LBP tidak hanya melihat hitam-putih huruf, tetapi memeriksa tekstur intensitas kecil di antara huruf dan di dalam goresan huruf. Ini menjelaskan mengapa hasil cetakan inkjet langsung terdeteksi berbeda dari letterpress.

Pada analisis visual peta variansi lokal, tepi huruf terlihat berpendar dengan pola acak. Letterpress tidak menunjukkan efek ini, karena tinta terdistribusi solid tanpa percikan halus.

Hasil ini sekaligus membantah klaim bahwa dokumen dicetak secara manual dengan handpress. Tidak ada bukti cetakan timbul, tidak ada tekanan fisik pada kertas, dan tidak ada tanda khas letterpress seperti garis tepi tinta yang lebih tebal akibat tekanan mekanis.

Metode NPA berbasis LBP sangat sulit untuk dimanipulasi, karena memanfaatkan properti intrinsik pola cetak yang tidak dapat diubah hanya dengan editing gambar. Jadi, jika hasil menunjukkan inkjet, sangat besar kemungkinan dokumen benar-benar dicetak dengan teknologi inkjet.

Ketiga area teks menunjukkan pola noise yang sama, artinya seluruh dokumen kemungkinan dicetak menggunakan satu perangkat printer digital, bukan gabungan metode cetak berbeda.

Keberulangan hasil analisis pada tiga bagian berbeda meningkatkan reliabilitas kesimpulan, sehingga kemungkinan kesalahan deteksi menjadi sangat kecil.

LBP_Energy yang sama pada ketiga analisis menunjukkan bahwa struktur tekstur cetakan konsisten, tidak ada bagian yang menyimpang signifikan. Ini mendukung asumsi bahwa satu metode cetak digunakan untuk seluruh halaman.

Apabila dicetak manual, hasil cetak biasanya memiliki ketebalan tinta bervariasi karena tekanan tangan yang berbeda, namun analisis tidak

menunjukkan pola tekanan semacam itu. Sebaliknya, hasil cetak terlihat homogen, tipikal printer modern.

Dari seluruh hasil, dapat disimpulkan bahwa metode forensik NPA berbasis LBP mendeteksi karakteristik cetakan inkjet dengan sangat jelas.

Kesimpulan ini tidak hanya berdasarkan satu indikator, tetapi kombinasi dari nilai LocalVar tinggi, pola granular pada LBP image, distribusi histogram LBP yang menyebar, dan ketiadaan ciri fisik letterpress.

Secara ilmiah, klaim bahwa dokumen dicetak manual dengan handpress tidak sesuai dengan bukti tekstur mikro yang terdeteksi.

Ciri cetakan inkjet seperti titik-titik halus dan variasi intensitas lokal terlihat di semua bagian teks, membantah klaim cetakan mekanis.

Tidak ada tanda khas letterpress berupa bekas tekanan atau cetakan timbul yang bisa diverifikasi melalui LBP atau visualisasi variansi lokal.

Dengan bukti teknis yang konsisten pada semua area yang dianalisis, kesimpulan paling kuat adalah bahwa dokumen tersebut dicetak dengan teknologi inkjet.

Hasil ini menegaskan keunggulan metode NPA berbasis LBP untuk memverifikasi keaslian dan metode pencetakan dokumen forensik secara objektif.

Secara keseluruhan, seluruh analisis mengarah pada satu kesimpulan: lembar pengesahan yang diklaim dicetak manual ternyata dicetak menggunakan printer inkjet.

REFERENSI

- R.H. Sianipar, "Java Referensi Lengkap untuk Programmer", 9789792965629, PENERBIT ANDI, 2019.
- R.H. Sianipar, "Matlab Untuk Pembelajaran Dan Riset Sinyal Digital", 9786230101984, PENERBIT ANDI, 2019.
- R.H.Sianipar, "Dasar Pemrosesan Citra Digital Dengan MATLAB", ISBN : 9789792966169, PENERBIT ANDI, 2018.
- R.H.Sianipar, "Sinyal Dan Sistem Dengan MATLAB", ISBN : 9789792967937, PENERBIT ANDI, 2018.
- R.H.Sianipar, "Komputasi Untuk Sains Dan Teknik Dengan Matlab", ISBN : 9789792965650, PENERBIT ANDI, 2018.
- R.H.Sianipar, "MATLAB Untuk Mahasiswa: Belajar dari Berbagai Studi Kasus", ISBN: 9789792953190, PENERBIT ANDI, 2017.
- R.H.Sianipar, "Struktur Data Dengan Pemrograman Generik", ISBN: 9789792947519, PENERBIT ANDI, 2015.
- R.H.Sianipar, "Pemrograman Java Untuk Programmer", ISBN: 9789792947823, PENERBIT ANDI, 2015.
- R.H.Sianipar dan Hamzan Wadi, "Pemrograman Python: Teori dan Implementasi", ISBN: 9786021514740, PENERBIT INFORMATIKA, 2015.
- R.H Sianipar, "Pemrograman MATLAB dalam Contoh dan Penerapan", ISBN: 9786028758994, PENERBIT INFORMATIKA, 2013.
- R.H Sianipar, "MATLAB untuk Pemrosesan Citra Digital", ISBN: 9786021514115, Penerbit INFORMATIKA, 2013.
- R.H.Sianipar, "C++ Untuk Programmer", ISBN: 9786021514108, PENERBIT INFORMATIKA, 2013.
- R.H.Sianipar, "Teori dan Implementasi JAVA", ISBN: 9786028758949, PENERBIT INFORMATIKA, 2013.
- R.H.Sianipar, "Pemrograman dan Struktur Data C", ISBN: 9786021514122, PENERBIT INFORMATIKA, 2013.
- R.H.Sianipar, "Java: Algoritma, Struktur Data, dan Pemrograman GUI", ISBN: 9786028758987, PENERBIT INFORMATIKA, 2013.
- R.H.Sianipar, "Pemrograman C++: Dasar Pemrograman Berorientasi Objek", ISBN: 9789792931846, PENERBIT ANDI, 2012.
- R.H.Sianipar, "Pemrosesan Sinyal Digital", ISBN: 9789792931013, PENERBIT ANDI, 2012.

DAFTAR PUSTAKA

1. Casey, E. (Ed.). (2010). *Handbook of digital forensics and investigation*. Academic Press.
2. Cornwall, H. (1985). *The hacker's handbook*. Century Publishing.
3. Houck, M. M. (Ed.). (2018). *Digital and document examination*. Academic Press.
4. Kelly, J. S., & Angel, M. (2021). *Forensic document examination in the 21st century*. Routledge.
5. Koppenhaver, K. M. (2007). *Forensic document examination: Principles and practice*. Humana Press.
6. Marcella, A., Jr., & Menendez, D. (2010). *Cyber forensics: A field manual for collecting, examining, and preserving evidence of computer crimes* (2nd ed.). Auerbach Publications.
7. Newman, R. C. (2007). *Computer forensics: Evidence collection and management*. CRC Press.
8. Hassan, N. A. (2019). *Digital forensics basics: A practical guide using Windows OS*. Apress.
9. Roy, A., & Derawi, M. O. (2018). *Digital image forensics: Theory and implementation*. Springer.
10. Tyagi, V. (2025). *Digital image forgery detection: Techniques, challenges and applications*. Springer.
11. Sammons, J. (2014). *The basics of digital forensics: The primer for getting started in digital forensics*. Syngress.
12. Altheide, C., & Carvey, H. (2011). *Digital forensics with open source tools*. Syngress.
13. Casey, E. (2011). *Digital evidence and computer crime: Forensic science, computers, and the internet* (3rd ed.). Academic Press.
14. Jones, A., Valli, C., Sutherland, I., & Thomas, P. (2006). *Building a digital forensics laboratory*. Syngress.
15. Nelson, B., Phillips, A., & Steuart, C. (2015). *Guide to computer forensics and investigations* (5th ed.). Cengage Learning.
16. Prosise, C., & Mandia, K. (2003). *Incident response & computer forensics* (2nd ed.). McGraw-Hill.
17. Reith, M., Carr, C., & Gunsch, G. (2002). *An examination of digital forensic models*. Lulu Press.
18. Watkins, K. L. (2019). *Practical forensic imaging: Securing digital evidence with Linux tools*. No Starch Press.
19. Rogers, M. K., & Seigfried-Spellar, K. C. (2017). *Digital crime and forensic science in cyberspace*. IGI Global.
20. Solanki, K., & Goyal, P. (2017). *Multimedia forensics and security*. CRC Press.

TENTANG BUKU

Buku ini menghadirkan beragam metode analisis, mulai dari uji lintasan stempel, Error Level Analysis (ELA), Local Binary Pattern (LBP), K-Means Color Clustering, hingga Multi-Scale Gradient Analysis. Setiap teknik forensik digital ini digunakan untuk memeriksa elemen penting dokumen seperti teks, tanda tangan, logo, foto, hingga stempel. Temuan-temuan yang dihasilkan menunjukkan adanya ketidakwajaran yang mengarah pada proses penyusunan digital, bukan pada metode cetak otentik.

Selain itu, analisis tipografi dan tekstur huruf memperlihatkan perbedaan jelas antara dokumen yang dicetak dengan mesin ketik atau letterpress dibandingkan dengan dokumen yang disusun menggunakan word processor modern. Konsistensi huruf, distribusi spacing, hingga pola cetak mikro mampu mengungkap indikasi kuat adanya rekonstruksi atau pemalsuan. Dengan pendekatan ini, forensik digital membuktikan perannya sebagai instrumen penting dalam autentikasi dokumen.

Akhirnya, buku ini menunjukkan bahwa kombinasi metode forensik digital dapat memberikan bukti visual, tekstural, dan matematis yang konsisten untuk membedakan dokumen asli dengan yang dimanipulasi. Melalui studi kasus seperti analisis ijazah Joko Widodo, karya ini memperlihatkan bagaimana sains forensik digital dapat menjadi alat verifikasi objektif, tidak hanya dalam kajian akademik, tetapi juga dalam pembuktian hukum dan sejarah.

BALIGE PUBLISHING



2025

